The Dissertation Committee for Mi Kyung Han
certifies that this is the approved version of the following dissertation:

Optimizing Opportunistic Communication in Wireless Networks

Committee:

Lili Qiu, Supervisor

Simon Lam

Yin Zhang

Gustavo de Veciana

Kang-Won Lee

Optimizing Opportunistic Communication in Wireless Networks

by

Mi Kyung Han, B.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2011

To Sunyoung, Yoonkee, Dongsu, Youngeun, Yoomin, and Solomon.

Acknowledgments

This dissertation would not have been possible were not for following people. This section provides a wonderful opportunity for me to thank them all.

First of all, I want to thank my advisor Lili Qiu. Lili has guided me throughout my entire Ph.D. career and she has been not only an excellent advisor but a friend, a sister, and an admirable role model. In each project I work on, she has inspired me to identify important and challenging research problems, guided me to take various approaches in solving them, and sharpened me until the completion of the project in pursuit of excellency and integrity. I not only have learned from what she says and suggests but also significantly from what she does. In short, her dedication, insights, guidance, and advice have been extremely valuable for me for the past six years and will continue to inspire me throughout my life.

Next, I would like to thank Yin Zhang, who has been providing significant downloads of insights and valuable comments for various projects I've been a part of. His dedication, brilliance, and a sense of humor have been a great fertilizer for my network research.

I would also like to thank all the other dissertation committee members: Simon Lam, Gustavo de Veciana, and Kang-won Lee for their valuable time and feedback regarding my thesis. I am very grateful to be able to access their expertise, which makes rich deposits to my knowledge bank in networks research.

It has been a great pleasure to work with my co-authors: Eric Rozner, Apurv Bhartia, Feng Wang, Ratul Mahajan, Upendra Shevade, Yi-Chao Chen, Vinoth Chandar, Han Hee Song, and You Suk Seung. In particular, Eric has been my

good friend throughout my Ph.D. years and the funniest person to work with. His contributions were not only to the projects but to people. His presence and leadership have been greatly beneficial to all of our lab members in forming a strong and happy research group. Apurv, although it is very difficult to pronounce his name correctly, has inspired me significantly for his undying passion for wireless research which is expressed through his hardwork and excellent results. Feng motivated my early Ph.D. years as I worked with him on interference modeling. Upendra is the kindest, smartest person I have ever met and I thank him for his dedication and the useful comments he provided. Han Hee Song has provided me with various useful advice in general which helped me move forward.

I would like to thank all the members in the Laboratory for Advanced Systems Research (LASR). In addition to Lili and Yin, I thank all the other wonderful professors, Lorenzo Alvisi, Mike Dahlin, Mike Walfish, and Emmett Witchel, and LASR alumni, Taewon Cho, Nalini Belaramani, Don Porter, Jean-Philippe Martin, Taylor Riche, Harry Lee, and Allen Clement, and many others. Their dedication altogether to cultivating LASR research culture of vibrant, energetic, interactive systems research has been the crucial stepping stone for me, and I see their investments have been producing good and lasting results.

I would like to thank my wonderful colleagues for their talents, brilliance, and friendship: Swati Rallapali, Ed Wong, Prince Mahajan, Manos Kapritsos, Tianji Li, Joyce Whang, Sangman Kim, Jiwoo Kim, Sangmin Lee, Gene Moo Lee and Jungwoo Ha. I admire Swati for being able to carry out tasks in a timely and excellent manner and for her wonderful personality. I thank Ed for bringing creativeness in our research environment. I thank Prince for his insightful comments and his pursuit of excellency motivates me. I thank Manos for introducing Greekness to my life and also for inspiring me with his smart Greek comments. I thank Tianji

for his intelligent observations and frankness which has sharpened the discussions during our lab meetings. I admire Joyce for her curiosity and passion for research. I thank Sangman and Jiwoo for brightening my day with their positive thinking and energy. I thank Sangmin for his warm personality, encouraging comments and advice. I thank Gene for his cheerful spirit and I admire him for this outstanding social networking skills as he commits himself into social networking research. I thank Jungwoo for his precious time listening to my unending questions and answering them in the most accurate ways. I wish them all the best in their success throughout their career and life.

Next, I would like to thank my family, Sunyoung Han (dad), Yoonkee Min (mom), Dongsu (brother), Youngeun Huh (sister-in-law), and Yoomin Han (nephew). Knowing my dad's dedication and love for me has been a firm foundation throughout my life. My mom's profound wisdom and priceless advice she gives have been crucial even in my wireless research life and without her powerful prayer I would have not made this it far. I also thank my brother Dongsu who is the most brilliant person that I know personally, for his valuable advice and genuine heart. I also thank Young-eun for standing strong in all circumstances and providing me a strong support in all times. I also thank and bless Yoomin for being born in May 7, 2011. He has been strongly motivating me to finish my dissertation early. I look forward to seeing him soon after I finish.

I would like to thank my mentors Abby Kim, Kwang-hyun Chung, Chongwoo Yu, Kelly Brown, Jon Doris, John Monger, Grace Moon, Gunyoung Lee, Minjeoung Kim, Hangil Kim and Mary Kim. Their advice has taken me to a new level, awakening me with various important life truths. Their wisdom, support, love, and prayer have been lifting me up in times of need.

I thank my dear friends who have become my family: Harrison Cho, Soon

hyong Kwon, Joon Choi, Ryan Chang, Susan Lee, Hwaheun Kim, Evon Lee, Joel Lee, Tony Kang, Eunhye Choi, David Moon, Caryn Werner, Courtney Werner, Lauren Nanson, Yookyung Lee, Meghan Strickland, Sarah Strong, Sarbrina Tsai, Kirok Choi, Kyoung-won Chang, Yongsub Lee, Kwangsun Hwang, Joonil Kim, Woori Pyun, Jongtaek Lee, Jougeun Yoo, Sook Kim, Racheal Ji, Jihye Sim, Yoonhee Kim, Sunghoon Ko, Hangyul Choi, Yerim Park, Grace Haeun Kim, Daanbee Kim, Haein Ji, and many others. I am thankful that this friendship and togetherness I had for the past six years will last throughout my life.

Also, I would like to thank all of my previous and current roommates: Songhee Han, Hyewon Byun, Seri Lee, Jiae Han, Mokeun Yoo, and Michelle Chae. I thank them for sharing their precious life with me.

Also, I thank my best friend Jimin Song who has been walking with me from early college years as we both committed our lives into computer science research. Without her I would have never made my college years, and also my Ph.D. years. She has been a strong supporter who understands me deeply in all circumstances and I also support her to finish her Ph.D. strong as well.

Finally, I would like to thank my dearest friend Jesus, who has given up his life for me so that I can have a new life and thus called me to be a computer scientist with a renewed and eternal purpose. In terms of my research life, he has been the first and the last author and the most generous and helpful reviewer of every single paper I wrote including this dissertation. He deserves all the honor and glory.

Optimizing Opportunistic Communication in Wireless Networks

Mi Kyung Han, Ph.D.
The University of Texas at Austin, 2011

Supervisor: Lili Qiu

Opportunistic communication leverages communication opportunities arising by chance to provide significant performance benefit and even enable communication where it would be impossible otherwise. The goal of this dissertation is to optimize opportunistic communication to achieve good performance in wireless networks. A key challenge in optimizing opportunistic communication arises from dynamic and incidental nature of communication. Complicated wireless interference patterns, high mobility, and frequent fluctuations in wireless medium make the optimization even harder.

This dissertation proposes a series of optimization frameworks that systematically optimizes opportunistic communication to achieve good performance in wireless mesh networks and vehicular networks. We make the following three major contributions:

First, we develop novel algorithms, techniques, and protocols that optimize opportunistic communication of wireless mesh network to achieve good, predictable user performance. Our framework systematically optimizes end-to-end performance (e.g., total throughput). It yields significant improvement over existing routing schemes. We also show that it is robust against inaccuracy introduced by dynamic network conditions.

Second, we propose a novel overlay framework to exploit inter-flow network coding in opportunistic routing. In this framework, an overlay network performs inter-flow coding to effectively reduce traffic imposed on the underlay network, and an underlay network uses optimized opportunistic routing to provide efficient and reliable overlay links. We show that inter-flow coding together with opportunistic routing and rate-limiting brings significant performance benefit.

Finally, we develop a novel optimization framework in vehicular networks to effectively leverage opportunistic contacts between vehicles and access points (APs). We develop a new mobility prediction algorithm and an optimization algorithm to determine an efficient replication scheme that exploit the synergy among Internet connectivity, local wireless connectivity, mesh network connectivity, and vehicular relay connectivity. Based on our framework, we develop a practical system that enables high-bandwidth content distribution and demonstrate the effectiveness of our approach using simulation, emulation, and testbed experiments.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation for Opportunistic Communication

Recently, wireless mesh networks and vehicular networks have emerged as attractive communication paradigm. They provide significant communication opportunities to users in diverse areas such as campus, residential, rural, and metropolitan areas. Unlike traditional wireline networks, these networks exhibit interference, high loss, high mobility, and link fluctuations. To combat such challenges and enhance user performance in these networks, a new way of communication is needed.

Opportunistic Communication: Traditionally, a sender commits to a specific node as the next-hop, and the traffic makes progress only when it reaches the selected next-hop. However, high loss rates in wireless networks make traditional routing inefficient. Moreover, in vehicular networks when there is no persistent end-to-end path between a source and its destination, traditional routing simply becomes infeasible.

Opportunistic communication effectively leverages communication opportunities arising by chance to provide significant performance benefits or even enable communication where it would otherwise be impossible. Below is a brief summary of how opportunistic communication can benefit the user performance in wireless mesh networks and vehicular networks.

1.1.1 Opportunistic Communication in Wireless Mesh Networks:

Wireless Mesh Networks: A wireless mesh network consists of nodes that can forward traffic for each other either directly or through a multi-hop path. It provides wireless broadband access to users in diverse areas such as personal, local, campus, rural, and metropolitan areas. Many cities across the world have deployed, or are planning to deploy wireless mesh networks [126,138,143,146,147]. Moreover, due to its cost effectiveness, they are being deployed in low-income communities, rural communities, and developing countries [51,138,144] where telecommunication services have been limited.

Opportunistic Routing in Wireless Mesh Networks: To combat wireless losses in the mesh networks, *opportunistic routing* has been proposed to exploit the broadcast nature of the wireless medium. Traditional routing uses a predetermined single best sequence of nodes between a source and its destination, committing the next-hop for a packet before transmission. In comparison, opportunistic routing allows any intermediate node that opportunistically receives the packet to cooperatively forward it toward its destination. Opportunistic routing improves throughput by taking advantage of transmissions that unexpectedly reached too far or too short and effectively combining multiple weak links into a stronger link.

1.1.2 Opportunistic Communication in Vehicular Networks:

Vehicular Networks: Vehicular networks have emerged from the strong need of increased road safety and communication on the move. There are projects across the world in various government, industry, and academia devoted to vehicular networks([19, 25, 58, 91, 94]). An increasing number of vehicles are now equipped with wireless communication devices, in-car sensors, and GPS systems. Vehicles

can communicate with each other or APs near the road side. Such networks can enable a variety of applications, such as road safety, emergency, traffic flow control, information retrieval, and media content sharing.

Leveraging Opportunistic Contacts in Vehicular Networks: In vehicular networks, communication happens opportunistically due to networks' unique features. First, there is no persistent end-to-end path between a source and its destination. Second, as vehicles move at a high speed, the contact between vehicles or between a vehicle and an AP is unplanned and short-lived (e.g., typically lasts for only a few seconds). Third, the movement of vehicles is constrained by the road and current traffic conditions, thus vehicles may make a sudden turn or stop and cause difficulty in predicting the trajectory of vehicles. In this case, traditional routing or mobile routing schemes that assume end-to-end connection and simple movement of nodes is not applicable. Therefore we have to leverage opportunistic contacts between vehicles and APs.

1.2 Systematic Optimization of Opportunistic Communication in Wireless Networks

Despite significant work in opportunistic communication and its optimization, how to *systematically optimize the end-to-end user performance* to achieve good wireless performance remains an open question. A key challenge in optimizing opportunistic communication arises due to the dynamic and incidental nature of such communication opportunities. Complicated wireless interference patterns, high mobility, and frequent fluctuations in wireless medium characteristics further exacerbate the problem.

In this dissertation, we present a series of novel optimization frameworks

that systematically optimize opportunistic communication to achieve good performance in wireless mesh networks and vehicular networks.

1.2.1 Optimization of Opportunistic Communication in Wireless Mesh Networks

There are two key factors that determine the performance of opportunistic routing in wireless mesh networks: (i) routes and (ii) rate limits. Routes represent the aggregate decisions on how much traffic a node should forward upon receiving a packet. Routes determine how effectively we combine available links to forward traffic and how efficiently we utilize network resources and exploit spatial reuse. Rate limits dictate how fast each traffic source injects traffic into the network. Rate limits should ensure that traffic source does not send more than what paths can support. Thus our optimization framework targets to jointly optimize routes and rate limits with the objective of optimizing the end-to-end user throughput in mesh networks.

We first develop a practical opportunistic routing protocol that achieves predictable performance. Then we explore inter-flow network coding to further enhance the performance.

1.2.1.1 Model-driven Optimization of Opportunistic Routing in IEEE 802.11 Mesh Networks

We present an accurate model-driven optimization framework of opportunistic routing in IEEE 802.11 mesh networks for both unicast and multicast traffic. The framework includes the following components:

First, we develop a novel optimization algorithm to jointly optimize rate limiting and opportunistic routing. We derive a set of opportunistic constraints to

probabilistically characterize the available communication opportunities. Second, we develop a simple, yet accurate model for the widely used IEEE 802.11 protocol to capture interdependency between sending rates, loss rates, and throughput on different links using $O(E)$ constraints, where E is the number of links. Our model can handle real-world complexities such as hidden terminals, non-uniform traffic, multi-hop flows, and non-binary interference. Third, we develop a non-convex optimization algorithm to find a local optimum solution. Our algorithm is flexible and can accommodate different objectives. Fourth, we develop a practical protocol by enforcing the opportunistic routes and rate limits computed by our optimization algorithm. Through extensive simulation and testbed experiments, we show that our model is accurate, and our protocol achieves significantly higher performance than state-of-the-art shortest path routing and opportunistic routing protocols. Furthermore, we evaluate the performance in dynamic and uncontrolled environments, and find it is robust against inaccuracy introduced by a dynamic network and consistently outperforms the existing schemes.

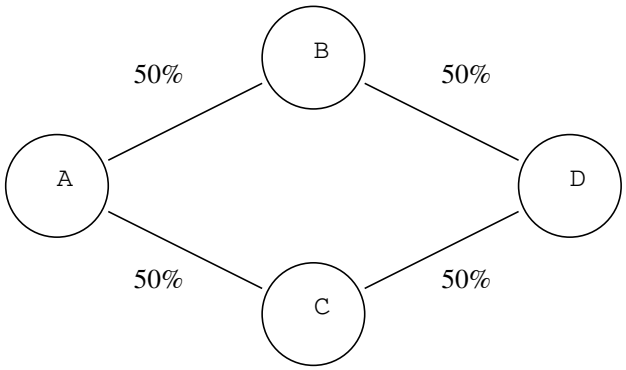


Figure 1.1: Motivating example for leveraging inter-flow network coding in opportunistic routing.

1.2.1.2 Overlay-based Optimization of Opportunistic Routing in IEEE 802.11 Mesh Networks

Inter-flow Network Coding: Inter-flow network coding is another approach in wireless mesh networks to improve network performance. In inter-flow network coding, an intermediate node combines multiple packets of different flows destined toward different next hops into one packet and broadcasts, thereby reducing the number of transmissions. We explore whether combining opportunistic routing with inter-flow coding can achieve further performance benefit.

Motivating Example: First, we demonstrate that inter-flow network coding can bring significant performance benefits to opportunistic routing by introducing the following example. Consider a diamond topology with two flows in Figure 1.1, where one flow goes from A to D and the other from D to A. *With traditional routing (no inter-flow or intra-flow coding)*, it takes two transmissions to deliver one packet over each hop due to a 50% loss rate, therefore, altogether 8 transmissions are required to deliver one packet for each of the two flows. *With opportunistic routing*, a flow source now utilizes both B and C as the forwarders. As long as either B or C receives a packet (instead of only B or only C), the traffic can make progress. Assuming independent packet loss rate on different links, which is reported in previous works [95], the probability of a packet reaching either B or C is 75%. Therefore, on average it takes only 1.33 transmissions to move a packet over the first hop (*i.e.*, deliver the packet to either of the intermediate nodes), and two transmissions to move the packet from the intermediate node to the destination. Therefore altogether 6.66 transmissions are required to deliver one packet for each of the two flows. *Inter-flow coding* [67] has benefits only if the two flows pick the same intermediate node as the forwarder. Assuming this, it still takes 2 transmissions to deliver one packet over each hop, the intermediate node can now XOR the

packets destined to A and D, and only make two transmissions to deliver packets to both receivers. Therefore, in the best case, it takes 6 transmissions to send one packet for the two flows. When the two flows use different forwarders, then there is no inter-coding opportunity, thus it takes 8 transmissions in total.

Interestingly, we observe that we can exploit inter-flow network coding to improve opportunistic routing. For the first hop, we use opportunistic routing, which takes only 1.33 transmissions. For the second hop, the intermediate node can XOR packets from the two flows whenever possible. In the *best case* where the intermediate nodes can XOR the packets received from each flow, the intermediate nodes only need 2 transmissions to deliver the XOR-ed packets for both flows. Thus it takes 4.66 transmissions in total to deliver one packet for each of the two flows. This yields a gain of 72% over single path routing, 43% over opportunistic routing alone, and 29% over inter-flow coding alone. In the *worst case* where the intermediate nodes cannot XOR any packets (which rarely occurs), it reverts to opportunistic routing and requires 6.66 transmissions, out-performing single path routing and (worst-case) inter-flow network coding by 20%.

Overlay-based Optimization of Opportunistic Routing Given the potential benefits of inter-flow network coding in opportunistic routing, we design our optimization framework to incorporate inter-flow network coding. A key challenge to achieve this goal lies in a strong tension between intra-flow coding used in opportunistic routing and inter-flow coding. While opportunistic routing tends to spread information across multiple nodes for higher reliability, such spread of information may reduce the decoding probability of inter-coded packet.

To address this challenge, we decouple inter-flow and intra-flow coding by proposing a novel framework where an overlay network performs overlay routing and inter-flow coding without worrying about packet losses, and an underlay net-

work uses optimized opportunistic routing and rate limiting to provide efficient and reliable overlay links. Based on this framework, we develop the first optimization algorithm to jointly optimize opportunistic routes, rate limits, inter-flow and intra-flow coding. We then develop a practical opportunistic routing protocol (O3) based on the optimization results. Using Qualnet simulation, we show that O3 significantly outperforms state-of-the-art routing protocols by simultaneously leveraging optimized opportunistic routing, inter-flow coding, and rate limits. Furthermore, we study the individual and aggregate benefits of opportunistic routing, inter-flow coding, and rate limits. Our results show that (i) rate limiting significantly improves the performance of all routing protocols, (ii) opportunistic routing is more beneficial under high loss rates, whereas inter-flow coding is more beneficial under low loss rates.

1.2.2 Optimization of Opportunistic Communication in Vehicular Networks

We observe that opportunistic communication plays a vital role in vehicular networks. We develop a novel optimization framework that leverages such opportunistic contacts between vehicles and APs to support high-bandwidth content distribution. To fully take advantage of the contacts between vehicles and APs, we proactively push content to the APs that the vehicles will likely visit in the near future. In this way, vehicles can enjoy the full wireless capacity instead of being bottlenecked by the Internet connectivity, which is either slow or even unavailable.

Our framework consists of the following key components: First, we develop a series of replication optimization techniques that enable the synergy among (i) Internet connectivity (which is persistent but has limited coverage and low bandwidth), (ii) local wireless connectivity (which has high bandwidth but short contact duration), and (iii) vehicular relay connectivity (which has high bandwidth but high

delay), and (iv) mesh connectivity among APs (which has high bandwidth but low coverage). Second, in order for the replication optimization algorithm to operate effectively, we develop a novel algorithm that predicts the set of APs a vehicle will visit in the near future with high accuracy. Third, we build a practical system that enables high-bandwidth vehicular content distribution (VCD). Using trace-driven simulation and Emulab-based emulation, we show that our novel replication scheme brings significant performance benefits compared to no replication and wireline or vehicular replication alone. The gap further increases as the ratio between wireless and wireline capacity increases. We further develop a full-fledge prototype VCD system in testbed that supports real video streaming running on smart phones and laptop clients with IEEE 802.11b and IEEE 802.11b networks. Our testbed results also confirm the effectiveness of our design.

1.3 Thesis Organization

The rest of the dissertation is organized as follows: In Chapter 2 we discuss previous work in wireless mesh networks, vehicular networks, and other work in relevant domain. In Chapter 3, we present an accurate model-driven optimization of opportunistic routing in IEEE 802.11 mesh networks. In Chapter 4, we present Optimization of Overlay-based Opportunistic routing (O3) in IEEE 802.11 wireless mesh networks. In Section 5, we present optimization framework for replication schemes that enable high-bandwidth Vehicular Content Distribution (VCD) in vehicular networks. Finally, we conclude our dissertation in Chapter 6.

Chapter 2

Literature Review

Opportunistic communication, due to its large potential performance benefit, has been extensively studied. In this chapter, we revisit prior research in opportunistic communication in wireless mesh networks and vehicular networks.

2.1 Related Work in Opportunistic Communication in Wireless Mesh Networks

We broadly classify related work in wireless mesh networks in the following categories: (i) traditional routing protocols, (ii) opportunistic routing protocols, (iii) analysis of opportunistic routing performance, (iv) wireless network modeling, and (v) network coding techniques used in wireless mesh networks.

2.1.1 Traditional Routing Protocols

Routing has been an active area in wireless networking research. A natural approach to routing traffic in multi-hop wireless networks is to adopt techniques similar to those in wired-line networks, which select a best path for each source-destination pair and send traffic along the predetermined path. Many of the existing routing protocols [40, 64, 101, 111–113] fall into this category, which we refer to as traditional routing.

DSDV [112] is based on classical Bellman-Ford routing and guarantees

loop-freedom. Every mobile node in the network maintains a routing table which records all possible destinations in the network along with the corresponding hop-count and the sequence number to avoid routing loops. Routing table updates are periodically transmitted throughout the network to maintain table consistency.

Different from the above approach, where a complete list of routes is maintained at each node, source-initiated on-demand routing [64, 111, 113] has been proposed. The source node initiates route discovery on demand, and only the nodes on the route participate in the route maintenance procedure until the route no longer exists or, no longer needed. In particular, AODV [113] builds on DSDV but minimizes the number of broadcast transmissions required for route discovery and maintenance by creating routes on demand and only maintaining the relevant routes. DSR [64] is another on-demand routing protocol based on source routing. Each node maintains route caches that contain the source route information, and the cache entries are continually updated. TORA [111] is a distributed routing protocol based on the concept of link-reversal. It focuses on establishing routes quickly and minimizes communication overhead by localizing algorithmic reactions to topological changes when possible. TORA often uses longer routes to avoid the overhead of discovering newer routes.

Along with the aforementioned routing protocols, various link-quality metrics have been proposed. AOVD and DSR use a metric based on hop-counts. However, research [10, 31, 38, 39] has found that using such metric does not provide good performance due to differing link quality across hops. They propose various alternate metrics taking into account link loss rate, packet transmission time, or signal-to-noise ratio. In particular, ETX [31] calculates the expected number of transmissions required to deliver a packet from a source to its destination using per-link measurements of packet loss ratio in both directions.

Since the above traditional routing protocols target high-mobility scenarios, they focus on finding and maintaining routes when there are frequent changes to the network. However, since nodes in wireless mesh networks have little or no mobility and sufficient computational power and energy, the primary focus of routing protocols in wireless mesh networks now becomes improving network performance.

2.1.2 Opportunistic Routing Protocols

As opposed to traditional routing, opportunistic routing has been proposed to improve performance. While traditional routing commits the next-hop for a packet before transmitting, opportunistic routing allows any intermediate node that opportunistically receives the packet to forward it toward the destination. Opportunistic routing effectively takes advantage of broadcast nature of the wireless medium and achieves higher performance gain than traditional routing.

ExOR [18] is the seminal opportunistic routing protocol. In ExOR, a sender broadcasts a batch of packets (*e.g.*, 10-100 packets per batch), and any forwarder receiving a packet is allowed to forward toward the destination. Since multiple forwarders may receive the same packet, forwarders need to coordinate to prevent duplicate transmissions. In ExOR, forwarders perform forwarding in the order defined by their proximity to the destination according to the ETX metric [31] and follow strict timing constraints to avoid redundant transmissions. Specifically, forwarders use a batch map, which records the list of packets they have received. A forwarder only forwards data that has not been acknowledged by the forwarders closer to the destination.

ROMER [153], another opportunistic routing protocol, tries to forward the packets simultaneously along multiple paths. It incorporates a credit-based scheme to limit the number of transmissions that a packet is allowed before reaching the

destination. Even with the credit-based scheme, there is still significant overhead since a packet is allowed to be forwarded by multiple nodes at each hop. Not only setting credits is non-trivial but also such static credits have difficulties in coping with different topologies.

Zhong *et al.* [160] show that the routing metric used to select and prioritize forwarding nodes is important. They develop a new routing metric, called EAX, to account for inter-candidate communication in opportunistic routing, and show that EAX outperforms ETX.

Ferriere *et al.* [41] also point out the limitation of using ETX for selecting forwarding nodes. In particular, they show that single-path metrics, such as ETX, ignore the nodes with low delivery rate to each of its neighbors even though collectively the links to multiple neighbors can form a strong wireless link. Based on this insight, they develop least-cost opportunistic routing, which quantifies the forwarding cost as the cost of reaching any neighbor in the direction towards the destination.

Recently, MORE [27] proposed a clever idea of applying intra-flow network coding to opportunistic routing that removes the need for fine-grained coordination among forwarders. Since random coding can effectively generate linearly independent coded packets with a high probability, the forwarding nodes in MORE do not need to coordinate which packets are forwarded by which nodes. Instead, in MORE each forwarding node just computes how much traffic it should forward and independently generates random linear combinations of all the packets it receives from the current batch. By removing the need for strict coordination, MORE can significantly outperform ExOR while supporting both unicast and multicast traffic.

There are a number of extensions that have been proposed to improve the performance of MORE. For example, CodeOR [82] improves MORE by transmit-

ting multiple batches per round-trip time to reduce overhead. SlideOR [83] proposes a scheme to encode data across multiple batches to reduce overhead, and CCACK [72] develops an acknowledgement scheme for downstream nodes to efficiently inform an upstream node which packets have been received.

All of the above opportunistic routing protocols optimize based on indirect metrics (*i.e.*, link quality, etc), rather than user-centric metrics (*i.e.*throughput, delay, etc). The major pitfall of such approaches is the poor correlation between the metric and user performance due to wireless interference and other complicated factors arising from the dynamic and incidental nature of wireless communication. Thus, minimizing the total cost based on indirect metrics does not always result in improved user end-to-end performance. Our framework systematically optimizes user-centric metrics. In addition, existing optimization approaches lack predictive power. They optimize based on current network condition and selects a path. However, when the path is actually used, the network conditions are changed due to interference, and the selected path consequently may not result in improved performance. Our framework is capable of predicting the outcome of the optimization and ensures the expected outcome is indeed achievable in a real network. Finally, these protocols focus on determining the forwarding paths, and place no restrictions on how fast each source can send. Letting the sources send as fast as they desire may result in significant performance degradation since the additional traffic that cannot be supported by the network only creates more interference. Our framework jointly optimizes routes and rate limits for wireless opportunistic communication.

2.1.3 Theoretical Analysis of Opportunistic Routing

There have been several studies on analyzing the performance of opportunistic routing. For example, Zeng et al. [155] develop a methodology for estimat-

ing maximum throughput given forwarding paths and traffic demands, and extends the work to multi-radio multi-channel wireless network [156]. Both studies assume the opportunistic routes are given, where each node only forwards the packets to be delivered toward the destination. Such selected routes are not optimal for two reasons: (i) as shown by Dubois-Ferrier et al. [41], prioritizing forwarders according to ETX is not optimal since the single path metric, like ETX, does not capture the anycast performance in the opportunistic routes, and (ii) due to wireless interference, we may sometimes prefer nodes farther away from the destination to forward traffic that has been received by nodes closer to the destination (in other words, none of least-cost path metrics can optimize end-to-end throughput).

A few studies (*e.g.*, [88, 118, 132, 137, 159]) propose optimization frameworks for opportunistic routing. These frameworks use convex (or linear) wireless network models and apply convex (or linear) programming to solve the resulting optimization problem. In contrast, we show that in order to achieve accurate network performance it is necessary to use an accurate network model that characterizes the non-convex relationship between the performance of different wireless links. This calls for an accurate wireless model and an efficient algorithm for searching for a close-to-optimal solution, which we address in our framework (Chapter 3). In addition, our work goes beyond theoretical analysis, which is the primary focus of the above work, by developing a practical routing protocol that realizes the performance gains in a real IEEE 802.11 network.

2.1.4 Wireless Network Modeling

Significant research has been done on wireless network modeling. One class of work focuses on asymptotic performance bounds. The seminal work by Gupta and Kumar analyzed the capacity of a wireless network under certain traffic pat-

terns and topologies [53]. Other researchers have since extended this work to other traffic patterns [78], mobility [52], and network coding [49]. These models provide useful insights as a network scales, but they cannot be applied to a specific network. Another large class of models predicts performance for a given scenario (*e.g.*, [17,46,48,66,116,120]). They differ in the generality of the model: some assume that everyone is within communication of each other [17,46,48,73], while others assume restricted traffic demands (*e.g.*, a single flow [46,48], two flows [120], sending to a single neighbor [47], or adding one new flow [123]). The two models [66,116], which can handle an arbitrary number of flows in a multihop network, are restricted to one-hop demands and have exponential complexity.

Furthermore, models in this class predict performance under a given scenario and cannot support optimization without enumerating all possible network configurations, which is prohibitive due to an intractable search space. To facilitate optimization, we need a model that can specify the entire region of feasible network configurations using a compact set of constraints, which can then be incorporated into the optimization procedure to optimize the desired objective within the feasible region. There are two models that fall into this category: (i) a conflict-graph based model proposed in [61] and (ii) an unicast interference model developed in [79]. The former assumes perfect scheduling, which does not hold in real wireless networks, so it tends to significantly over-predict the real performance, as we show in Section 3.8. The latter works well for the purpose of rate-limiting unicast flows but has difficulties meeting the needs of optimizing opportunistic routing, which requires accurate estimation of every link's performance under diverse traffic loads.

2.1.5 Network Coding

Wireless network coding has been proposed to enhance the efficiency of a packet transmission by exploiting the broadcast nature of the wireless medium.

The value of coding in routers is first acknowledged by Ahlswede et al. [3], in which they include theoretical bounds on the capacity of networks. The combination of later work [59, 70, 77] shows that linear codes achieve the maximum capacity bounds in multicast traffic, and coding and decoding can be done in polynomial time. Additionally, Ho et al. [56] show the above still holds when the routers pick random coefficients.

Moreover, there are studies that give theoretical analysis on network coding in wireless and wired networks [60, 89]. In terms of protocol implementation, Park [110] and Widmer at al. [142] provide simulation-based evaluation of the designed protocol, and Chachulski et al. [27] provide a real protocol implementation and evaluation using mesh network testbed.

For inter-flow coding, COPE [67] presents a practical routing protocol using inter-flow network coding for unicast in multi-hop wireless networks. It lets an intermediate node combine multiple packets towards different destinations into one packet, thereby reducing the number of transmissions and improving throughput. There are many follow-up works that enhance COPE. First, there are studies that develop techniques to select routes that create more coding opportunities [34, 74, 128]. Second, a few work jointly optimize network coding and scheduling [28, 124]. Third, a few work studies the impact of the data rates on the network coding. In particular, Kim et al. [69] pick the modulation rate that takes into account both coding gain and data rate, and Yun et al. [154] propose a technique to XOR packets that have different modulations.

Researchers have mainly focused on applying inter-flow network coding to traditional routing, where the routes are known before packet transmissions. Harnessing the benefit of inter-flow coding in opportunistic routing is more challenging due to uncertainty in the final routes being selected. There have been a few preliminary attempts that try to exploit inter-flow network coding in opportunistic routing, as evidenced by a few short papers [71, 151, 157]. They focus only on one aspect of the routing design: among multiple nodes that receive the data, which one to pick to actually forward the data. They use EXOR-style opportunistic routing and impose strict forwarding order, which requires significant coordination and spatial reuse. Only [115] considers the use of intra-flow coding as in MORE to avoid duplicates without coordination. However, it recognizes the significant challenges of applying inter-flow coding to general opportunistic routing, so it only supports opportunistic receptions over a single path, which significantly reduces efficiency under lossy links. Moreover, it does not develop a routing protocol and only uses numerical estimation of the number of transmissions based on the assumptions of 1-packet flow and perfect acknowledgement, which makes comparison hard.

In short, the existing works have four major limitations. First, they use pre-existing opportunistic routing protocols to route their data and do not select their opportunistic routes in an inter-flow coding-aware manner. Second, these heuristics try to reduce the number of transmissions but do not directly optimize end-to-end performance. The number of transmissions has been shown to have limited predictive power on end-to-end performance [67, 80]. In particular, COPE [67] shows that even in a simple 3-hop topology the coding gain (*i.e.*, the reduction in the number of transmissions) is very different from the MAC gain (*i.e.*, the improvement in throughput). Third, in order to limit the overhead of opportunistic routing, they restrict the forwarding nodes that are close to one another, limiting the inter-flow

coding opportunities. Fourth, their evaluations either use toy topologies ([71, 157]) or only compare with COPE ([151]) thus more comprehensive study comparing the benefits of various routing protocols using general topologies is needed.

Summary: Due to unpredictable and unreliable wireless medium, how to optimize opportunistic routing to maximize user performance is still an open problem. To address this, it is necessary to develop a systematic optimization for end-to-end user performance that captures the effects of all these components on network performance. To this end, we develop the first optimization framework that jointly optimizes routes, rate-limits and coding opportunities, and we design a practical opportunistic routing protocol based on our framework. Furthermore, we conduct extensive evaluation to compare a diverse set of routing schemes in order to demonstrate the effectiveness of our design.

2.2 Related Work in Opportunistic Communication in Vehicular Networks

We classify related work in opportunistic communication in vehicular networks into following categories: (i) vehicular network architecture and protocols, (ii) disruption tolerant networks, and (iii) mobility prediction techniques and prefetching.

2.2.1 Vehicular Networks

The seminal vehicular network architecture was proposed in the Infostations project to provide low-cost, high-bit rate connections to vehicles and pedestrians, enabling content delivery over a discontinuous area of high-bandwidth wireless coverage [45].

With the recent prolification of WiFi technology, researchers have conducted more realistic experimental studies in vehicular WiFi access. Drive-thru Internet project [106–108] presented experimental results with a single car driving past a single AP and proposed a session protocol which provides persistent end-to-end communication even in the presence of intermittent connectivity. More recently, the CarTel [26] project investigated general architectures for vehicular sensor networks and proposed that APs deployed in cities can be used as an uplink network for moving cars. The UMass DieselNet is a disruption-tolerant network deployed in Amherst on 40 buses each equipped with a Diesel Brick containing network interfaces and GPS device. Using DieselNet, extensive measurements studies have been conducted regarding mobility, bus-to-bus and AP-to-bus connectivity [15, 158], and new bus-to-bus routing protocols [11, 20] have been designed.

Moreover, as the current vehicular WiFi architecture suffers from short connection time, high loss rates, and intermittent connectivity, many techniques have been proposed to optimize network performance despite these adverse conditions. Cabernet [43], built on the Cartel project, addresses that current stock implementations of wireless networking protocols are not well suited for vehicular applications. To address this, Cabernet uses three techniques to obtain high throughput. First, to maximize the usable connection time between an AP and a vehicle they developed QuickWiFi, which achieves faster association and optimal scanning. Second, to improve end-to-end throughput over lossy wireless links, they developed the Cabernet Transport Protocol (CTP) which distinguishes wireless loss from congestion loss. Finally, to improve link quality they propose static bit-rate selection. Moreover, Camp et al. [24] conduct an in-depth study of various rate adaptation schemes in vehicular networks and proposes to select data rates based on a combination of RSSI and channel coherence time. Navda et al. [102] use directional antennas to

maximize the transfer opportunity between the vehicle and the AP.

In addition, in order to support realistic applications such as VoIP and Web browsing, techniques to improve WiFi handoffs and alleviate disruptions in vehicular networks have been proposed [13, 131]. For example, ViFi [13] takes advantage of multiple APs to minimize disruptions in WiFi handoffs and exploits opportunistic receptions by nearby APs. In their opportunistic reception scheme, one anchor AP is backed up by several auxiliary APs nearby, and when auxiliary APs opportunistically overhear a packet that is not acknowledgement, they probabilistically relay the packet to the intended next hop. On the standards track, IEEE 802.11r [131] is implementing extensions to IEEE 802.11 [105] to directly support fast handover in the MAC protocol, specifically targeted for vehicular environments where a mobile device moves from one AP to another in a matter of seconds. 802.11r minimizes the number of transition messages to 4 by piggybacking the security and QoS related messages with 802.11 authentication and reassociation messages.

These works are complementary to our work in that they focus on optimizing one-hop communication between a vehicle and nearby APs, while we focus on end-to-end performance. Therefore we can potentially leverage these approaches to improve the performance of the last hop. With these enhancements, the gap between Internet and wireless capacity will further increase and makes replication even more important.

Recent proposals also include changing applications to cope with vehicular networks. In [12, 14], aggressive prefetching is used to make the results from web search queries available to mobile clients in buses. In particular, Thedu [12] transforms interactive Web search into a one-shot request/response process to reduce access delay. While these approaches require connecting with a remote server, our VCD framework replicates content to APs to eliminate the Internet bottleneck.

Finally, there has been some work on vehicle-to-vehicle communication. For example, SPAWN [33] uses gossip for file transfer and CarTorrent [75] extends SPAWN and is implemented in a testbed. [29] treats vehicular networks as a special type of DTN and focuses on leveraging vehicle-to-vehicle (V2V) communication to deliver content.

Inspired by the analysis in [15], we leverage APs as rendezvous points for replicating content among vehicles. We focus on optimizing content replication given limited wireline and wireless resources, which has not been previously explored.

2.2.2 Disruption Tolerant Networks

Vehicular networks can also be considered as a special type of disruption tolerant networks. Different from traditional DTNs, which focus on communicating with a specific node, we focus on content delivery.

Most DTN routing protocols perform epidemic routing, *i.e.*, to replicate packets at each transfer opportunity hoping to find a path to a destination. In epidemic routing, various methods to limit replication or to clear useless packets have been proposed to avoid wasteful resource usage. Some work proposes using historic meeting information [20,21,35,84] use historic meeting information. A few studies replicate packets with small probability [140]. Also some work bounds the number of replication per packet [63,134,141]. Other studies [11,65] estimate best contacts according to some utility function and only replicate on best contacts. Moreover, there are approaches using network-coding [62, 142] to increase efficiency and to add redundancy to cope with failures. Also, there are techniques to infer already delivered packets and remove them [20, 130].

Much work in DTN routing assumes unrealistic resource availability. Some [63,

76,134,141] assume unlimited storage and unlimited bandwidth, and others [21,35, 130] assume limited storage but unlimited bandwidth. While they provide valuable insights in theoretical tractability, they are impractical to use in realistic scenarios. Different from these theoretical approach, a few studies [11, 20] propose DTN protocols with finite storage and bandwidth assumption and uses vehicular DTN traces.

The aforementioned DTN routing protocols use a variety of mechanisms, including discovering the meeting probabilities among nodes, packet replication, and network coding. The primary focus of these mechanisms is to increase the likelihood of finding a path with limited information, so these approaches only have an incidental effect on routing metrics such as maximum or average delivery delay. For example, RAPID [11] explicitly tries to optimize system-wide metrics such as average delay while incorporating resource constraints. Our approach leverage both utility-based optimization and epidemic replication to achieve efficiency and robustness.

2.2.3 Mobility Prediction

To provide seamless and ubiquitous connectivity to mobile users, cellular networks researchers have studied mobility prediction or mobility tracking [6, 7, 16, 81, 85, 152]. In [1], the authors assume that a mobile terminal take shortest paths when they move from one cell to another with four possible direction and calculates its location probability at the time of a call arrival. In [85], a hierarchical location-prediction algorithm is proposed, where a two-level mobility model is used to represent the movement behavior at global and local levels, and the next cell is predicted based on speed and direction of a user's trajectory. In [16] the next probable cell is determined based on path information. In [7], rather than a single

cell, a group of future cells that a mobile terminal will most likely move to is determined by estimating the user's trajectory and arrival and departure times. [6] proposes a more comprehensive framework, User Mobility Profile (UMP), which consists of historic records and predictive patterns of mobile terminals. It uses an adaptive prediction algorithm to predict a group of cells that a mobile terminal will move to by considering historical records, path information, moving direction and seed, and cell residence time.

There is also a large body of mobility prediction literature in the context of WiFi networks. Ghosh et al. [50] build mobility profiles for users and statistically predicts the next social hub the user will visit. Nicholson et al. [103] build the user's customized mobility models on the devices themselves and use a second-order Markov model to predict the connection opportunity and its quality of the device with an access point. McNamara et al. [92] use the past history to identify opportunities for media sharing in ad-hoc DTNs. Song et al. [133] compare various predictors in literature using real data and suggest that second-order Markov model with a simple fallback mechanism (when there is no prediction) performs well. These existing algorithms optimize for non-vehicular environments (with lower node speeds, higher location update frequency, and constant update intervals) and thus may not be as effective in vehicular networks, since vehicles travel much faster. Moreover, the GPS update frequency tends to be relatively low (e.g., once per minute), and the inter-arrival time between two consecutive updates may not be constant. Thus vehicular networks faces new challenges in mobility prediction. [37] specifically targets vehicular networks. It shows that as people frequently drive on familiar routes on a regular basis, mobility and connectivity related information along their routes can be predicted with good accuracy using historical information, such as GPS tracks with timestamps, RF fingerprints, and link and network-layer

addresses of visible APs. However, collecting historical data to extract a pattern requires additional measurement overhead and consent from users to collect such data, raising privacy concerns.

Summary: Most existing work has following limitations: First, they focus exclusively on protocol-level optimization of one-hop communication. Second, they rely on heuristics to guide data replication. Third, they completely ignore resource constraints such as Internet bandwidth, wireless bandwidth, etc. As opposed to existing approaches, our VCD framework provides a general and flexible optimization approach that enables high bandwidth content distribution. We specifically formulate a linear program to maximize the user satisfaction under the predicted mobility pattern and traffic demand subject to the resource constraints. Furthermore, based on the framework, we develop a practical vehicular content distribution protocol. By simulation, Emulab and testbed implementation, we demonstrate the effectiveness of our VCD framework.

Chapter 3

Model-Driven Optimization of Opportunistic Routing in Wireless Mesh Networks

3.1 Introduction

Opportunistic routing has been proposed to exploit communication opportunities that arise by chance due to the broadcast nature of the wireless medium. When a sender broadcasts its data, any node that hears the transmission may forward the data toward the destination. Although individual nodes may experience high loss rates, as long as there exists one forwarder that is closer to the destination and receives the transmission, the data can move forward. In this way, opportunistic routing can effectively combine multiple weak links into a strong link and take advantage of transmissions reaching unexpectedly near or unexpectedly far.

There are two key factors that determine the performance of opportunistic communication in wireless mesh networks: (i) routes (*i.e.*, for a given flow how much traffic node j should forward upon receiving a packet from another node i), and (ii) rate limits (*i.e.*, how fast each traffic source can inject traffic into the network). Routes determine how effectively we take advantage of communication opportunities and how efficiently we utilize network resources and exploit spatial reuse. Rate limits ensure that traffic sources do not send more than what paths can support. Without appropriate rate limits, the network throughput can degrade drastically under traditional shortest-path routing [79]. Rate limiting is even more critical for opportunistic routing due to its use of broadcast transmissions: (i) broadcast

transmissions do not perform exponential backoff (*i.e.*, its contention window does not increase upon packet losses) and thus are more likely to cause network congestion; and (ii) broadcast transmissions preclude the use of 802.11's synchronous ACK mechanism, and receivers' feedback has to be sent above the MAC layer, which can easily get lost during network congestion and cause unnecessary retransmissions and serious throughput degradation.

In this chapter, we describe an accurate model-driven optimization framework that jointly optimize routes and rate limits for opportunistic communication. We focus on static, 802.11-based, multihop networks, though we believe that the general methodology is applicable to other scenarios. We develop the first opportunistic routing protocol that can accurately optimize IEEE 802.11 end-to-end performance (*i.e.*, the performance derived from optimization can be realized in a real IEEE 802.11 multihop network). This distinctive feature is important given the wide deployment of IEEE 802.11 networks.

Challenges: Accurate optimization of opportunistic communication in an IEEE 802.11 network is challenging for the following four reasons. *First*, the dynamic and incidental nature of communication opportunities makes it difficult to estimate their impact on the resulting network performance. *Second*, optimization of opportunistic routing places stringent requirements on a network model: the model should (i) specify the region of feasible network configurations using a compact representation so that we can optimize the objective within the feasible region as defined by these constraints, (ii) accurately estimate performance on every link in the network (as opposed to only a small number of links on specified routes, as in [79], for the purpose of optimizing rate limiting alone), and (iii) be accurate across a wide range of traffic conditions, including high traffic load, which is common in opportunistic routing. *Third*, the non-convex interference relationships among different links

and the huge search space of possible opportunistic routes and rate limits impose significant challenges on the optimization procedure itself. *Fourth*, to be valuable in practice, the resulting optimization solution should be easy to implement, using only a small number of control knobs.

Approach and contributions: We address the above challenges using the following four steps:

1. *General optimization framework.* We develop a general framework to jointly optimize routes and rate limits for opportunistic communication (Section 3.2). The framework uses opportunistic constraints to probabilistically characterize the available communication opportunities. It can use different wireless interference models.

2. *Interference model for IEEE 802.11 broadcast traffic.* The complex interference, traffic, and MAC-induced dependencies in the network are often the underlying cause of unexpected behavior. We develop a new model to capture these dependencies for broadcast transmissions (Section 3.3). We use measurements from a given network to estimate link loss rate, carrier sense probability, and conditional collision loss probabilities to seed our model. Our model derives the relationships between sending rates, loss rates, and throughput to capture the effects of carrier sense and collisions. Our model involves only $O(E)$ constraints, where E is the total number of edges. Thus it can be easily incorporated into our optimization framework. Despite its simplicity, the model captures real-world complexities such as hidden terminals, non-uniform traffic, multihop flows, non-binary and asymmetric interference.

3. *Iterative procedure for non-convex optimization.* Since our model is non-convex, we develop an iterative optimization procedure to find a local optimal solution

(Section 3.4). Our algorithm is flexible and can accommodate different performance objectives. For comparison, we explore an alternative approach that uses a widely used conflict-graph-based interference model [61] that is less accurate [79, 116], but convex, and thus allows global optimization. Our results show that our approach of combining a more accurate model with non-convex optimization yields better and more accurate performance.

4. *Practical installation of routes and rate limits.* We develop a practical opportunistic routing protocol that implements the opportunistic routes and rate limits optimized by our algorithm in real networks (Section 3.5). The mechanisms for installing routes and rate limits can support both unicast and multicast.

We implement our protocol in both the Qualnet simulator [117] and a 21-node wireless mesh testbed using Click [30] and the MadWiFi driver [90]. Extensive simulations and testbed experiments (Section 3.7– 3.9) show that our approach achieves high accuracy (*i.e.*, the difference between the achieved performance and our model estimation is within 20%) and significantly out-performs state-of-the-art shortest path and opportunistic routing protocols (*e.g.*, its total throughput is up to 14x ETX's throughput and 11x MORE's throughput). We further study the impact of dynamic and uncontrolled environments on accuracy and performance, and find that our approach is robust to inaccuracy in the input and it also consistently out-performs the existing schemes (Section 3.10).

3.2 Optimization Framework

Overview: We develop a general framework for jointly optimizing opportunistic routes and rate limits. Our formulation assumes the use of network coding, which prevents nodes from forwarding redundant information without requiring

$Flows$	the set of unicast or multicast flows
$src(f)$	source of flow f
$dest(f, d)$	d-th destination of flow f
$Demand(f)$	traffic demand of flow f, *i.e.*, the amount of traffic f desires to send
$G(f)$	throughput of flow f
$T(f, i)$	node i's sending rate for flow f
$Y(f, d, i, j)$	information receiving rate along link $i \rightarrow j$ for d-th destination in flow f ($d = 1$ for unicast)
$P(i, j)$	loss rate of link $i \rightarrow j$ (including both collision and inherent wireless medium loss)
$\mathcal{N}(i)$	a subset of i's neighbors
$S(i, \mathcal{N}(i))$	success rate from node i to i's neighbor set $\mathcal{N}(i)$

Table 3.1: Notations for optimizing opportunistic routing.

fine-grained coordination among different nodes. Without loss of generality, we focus on multicast flows, since unicast flows are a special case of multicast with one receiver in each multicast group. The main design issue becomes how fast each traffic source should send traffic and how much traffic an intermediate node should forward to achieve high performance. This can be formulated as an optimization problem that maximizes total network throughput subject to information conservation constraints, opportunistic constraints, and interference constraints. Figure 3.1 shows the resulting formulation, and Table 3.1 specifies the variables in the formulation.

Optimization objective: Given the set of unicast or multicast flows $Flows$, and the traffic demands $Demand(f)$, our optimization outputs traffic sending rates $T(f, i)$ and information receiving rates $Y(f, d, i, j)$, which will be converted to opportunistic routing configurations using a credit-based scheme described in Section 3.5. As shown in Figure 3.1, the first term in the objective, $\sum_{f \in Flows} G(f)$, reflects the pri-

$\triangleright\ Input:\ Flows,\ Demand(f)$
$\triangleright\ Output:\ T(f,i),\ Y(f,d,i,j)$
maximize: $\sum_{f \in Flows} G(f) - \beta \sum_{f,i} T(f,i)$
subject to:

[C1] $G(f) \leq Demand(f)$ $(\forall f)$

[C2] $G(f) \leq \sum_k Y(f,d,k,dest(f,d))$ $(\forall f,d)$

[C3] $Y(f,d,k,src(f)) = 0$ $(\forall f,d,k)$

[C4] $Y(f,d,dest(f,d),k) = 0$ $(\forall f,d,k)$

[C5] $\sum_k Y(f,d,k,i) \geq \sum_j Y(f,d,i,j)$
$(\forall f,d,i : i \neq src(f)$ and $i \neq dest(f,d))$

[C6] $S(i,\mathcal{N}(i))T(f,i) \geq \sum_{k \in \mathcal{N}(i)} Y(f,d,i,k)$ $(\forall f,i,\mathcal{N}(i))$

[C7] $interference\ constraints\ on\ T_i \triangleq \sum_f T(f,i)$

Figure 3.1: Problem formulation to optimize multicast throughput of opportunistic routing.

mary goal of maximizing the total throughput over all flows. The second term in the objective, $-\beta \sum_{f,i} T(f,i)$ represents the total amount of wireless traffic. Including both terms reflects the goals of (i) maximizing total throughput and (ii) preferring the least amount of traffic among all solutions that support the same total throughput (*e.g.*, avoiding loops and unnecessary traffic). Since the first objective is more important, we use a small weighting factor $\beta = 10^{-5}$ for the second term just for tie breaking (*i.e.*, only when the first objective is the same, we prefer the one with the least traffic).

To compute the first term, for a unicast flow f, $G(f)$ is its throughput. For a multicast flow f, $G(f)$ is the throughput of the bottleneck receiver. Note that there are many other ways to define the objective in multicast setting [129]. Here we use one of the metrics as an example. Our optimization framework can support other multicast objectives, such as total throughput over all receivers in the multicast group or other weighted versions. Moreover, while we focus on total

throughput, our framework can be directly applied to optimizing other objectives. For example, our evaluation also considers optimizing a linear approximation of proportional fairness, defined as $\sum_{f \in Flows} \log G(f)$, which strikes a good balance between fairness and throughput [119]. We can also maximize total revenue if the revenue of a flow is a function of its throughput.

Throughput constraints: To ensure $G(f)$ is the throughput of flow f, it has to satisfy constraints (C1) and (C2) in Figure 3.1. Constraint (C1) indicates that the throughput of a flow should be no more than its traffic demand (*i.e.*, total amount of information a source desires to send). Constraint (C2) ensures that $G(f)$ is no more than the total amount of information delivered from all links incident to the destination of flow f. For a multicast flow f, $G(f)$ should be no more than the total amount of information delivered to each destination in the flow f. Note that we do not need a lower bound on $G(f)$ since the objective is to maximize $G(f)$.

Information conservation constraints: To handle lossy wireless links, we distinguish traffic and information sent along a link. A feasible routing solution should satisfy information conservation. This property is given by constraints (C3–C5) in Figure 3.1. Constraint (C3) ensures no incoming information to a traffic source, constraint (C4) ensures no outgoing information from a destination, and constraint (C5) represents flow conservation at an intermediate node i, *i.e.*, the total amount of incoming information is no less than the total amount of out-going information.

Opportunistic constraints: Opportunistic routing exploits the wireless broadcast medium by having different nodes extract information from the same transmission. We formally capture this notion using opportunistic constraints, which relate traffic volume to the amount of information delivered.

For ease of explanation, we first consider one sender sending to two re-

ceivers, and then generalize it to an arbitrary number of receivers. Consider a sender s, and denote the link loss rates from s to its neighbors r_1 and r_2 as $P(s, r_1)$ and $P(s, r_2)$, respectively. It is evident that for a given flow the amount of information delivered to a neighbor is bounded by the product of the sending rate and link delivery ratio. Therefore we have $(1 - P(s, r_1))T(f, s) \geq Y(f, d, s, r_1)$ and $(1 - P(s, r_2))T(f, s) \geq Y(f, d, s, r_2)$. In addition, since there is overlap between the information delivered to r_1 and r_2, we are only interested in the non-overlapping information (*i.e.*, when redundant information is delivered to both nodes, it should only count once). The total non-overlapping information delivered to r_1 and r_2 should satisfy the following constraints:

$$(1 - P(s, r_1)P(s, r_2))T(f, s) \geq \sum\nolimits_{i \in \{1,2\}} Y(f, d, s, r_i),$$

where the left hand-side represents the total amount of traffic successfully delivered to at least one of the receivers, and the right hand-side represents the total non-overlapping information delivered to the receivers.

Now we consider a general setting, where a sender s has N neighbors. We enumerate all possible subsets of its neighbors. For each neighbor set $\mathcal{N}(i)$, we require:

$$S(i, \mathcal{N}(i))T(f, i) \geq \sum\nolimits_{k \in \mathcal{N}(i)} Y(f, d, i, k), \tag{3.1}$$

where $S(i, \mathcal{N}(i))$ denotes the delivery probability from i to at least one node in $\mathcal{N}(i)$. When delivery rates of different links are independent, which holds for some networks [135], $S(i, \mathcal{N}(i)) = 1 - \prod_{k \in \mathcal{N}(i)} P(i, k)$. When the link delivery rates are correlated, we can empirically measure $S(i, \mathcal{N}(i))$. Equation 3.1 indicates the total traffic successfully delivered to at least one neighbor in $\mathcal{N}(i)$ should be no less than the total non-overlapping information delivered to $\mathcal{N}(i)$. This results in (C6) in Figure 3.1. When i has many (say, K) neighbors, we limit the number of

such constraints by only enumerating neighbor sets of size 1, 2, and K (*i.e.*, we enumerate only O(K^2) instead of O(2^K) neighbor sets).

Interference constraints: Wireless interference has a significant impact on wireless network performance. In particular, nearby senders carrier sense and defer to each other. Moreover, since carrier sense is not perfect, there may be multiple overlapping nearby transmissions that can cause collisions. These effects can further constrain the amount of traffic on each link and introduce strong inter-dependency between sending rates, loss rates, and throughput. We address this issue in Section 3.3 by developing the constraints that capture the relationships between $T(f, i)$ and $P(i, j)$.

3.3 Broadcast Interference Model

3.3.1 Motivation for a Better Model

Despite significant research on modeling the impact of wireless interference, none of the existing models directly fulfills our need for optimizing opportunistic routing. To support optimization, we need a model that specifies the feasible region of network configurations using a compact representation. The following two existing models fall into this category.

Conflict-graph model: The first model, proposed in [61], is a conflict-graph model that represents wireless links as vertices and draws a conflict edge between two vertices if the corresponding wireless links interfere. Based on this definition, it is clear that links corresponding to an independent set in the conflict graph can be active simultaneously. Therefore, the interference constraints are the schedule restrictions imposed by the independent sets, which can be expressed as a set of linear constraints.

There are two limitations in applying the conflict-graph model for optimizing opportunistic routing. First, the model in [61] assumes perfect scheduling, *i.e.*, packet transmissions at different nodes can be precisely controlled and it overestimates the performance in real networks as we will show in Section 3.8. Second, the conflict-graph model is a link-based model, while opportunistic routing uses broadcast transmissions and requires a node-based broadcast model. Existing broadcast extensions of the conflict-graph model provide only an aggregate answer of whether two broadcast transmissions interfere or not. For example, some extensions [118, 127, 155, 159] conservatively consider two broadcast transmissions to interfere if any one of their receivers is interfered by the other transmission, while other extensions [155] consider broadcast transmissions to interfere if all of their receivers are interfered by the other transmission. A single aggregate answer on whether broadcast transmissions interfere does not fully characterize the impact of interference on different receivers and is therefore inadequate for use in optimizing opportunistic routing.

IEEE 802.11 unicast model: The other model, proposed in [79], models interference among unicast transmissions in IEEE 802.11. Since opportunistic routing uses broadcast traffic, we need to develop interference models for broadcast transmissions. Furthermore, as broadcast transmissions does not perform binary backoff to limit the sending rate (*i.e.*, its contention window does not increase even under packet losses), it is necessary to have an accurate model even for high traffic load and channel occupancy, which induces high collision losses, and the linear approximation used in [79] becomes inaccurate under high collision losses. In addition, [79] is used for rate limiting unicast transmissions when given specified routes. Therefore it suffices to accurately estimate the sending rates and loss rates on a small number of links used for routing. In contrast, for the purpose of route

optimization, we need to accurately estimate the performance for *all* receivers of a given sender, which is much more challenging.

Modeling goals and strategy: We develop our model specifically for IEEE 802.11 broadcast traffic. We observe that wireless interference affects IEEE 802.11 traffic in two important ways: (i) nearby senders cannot transmit simultaneously due to carrier sense, and (ii) transmissions may sometimes result in collisions due to imperfect carrier sense. We model these effects by developing the relationships between sending rates, loss rates, and throughput, which can be incorporated into our optimization framework and facilitate model-driven optimization. While our work applies the model to optimizing opportunistic routing, the model is useful in other contexts (*e.g.*, optimizing network topology and network planning). Our model is general and captures real-world complexities (*e.g.*, hidden terminals, multihop flows, non-binary interference, and heterogeneous traffic), which is confirmed by simulation and testbed experiments using multihop networks in Section 3.8. Compared with [79], both our sender model (Section 3.3.3.1) and loss model (Section 3.3.3.2) are much more refined and do not involve any linear approximation. Thus, our model can more accurately estimate the loss rates for all receivers even under heavy traffic loads, which is essential for the optimization of opportunistic routing.

3.3.2 Background and Assumptions

We first review the broadcast transmissions as specified by the IEEE 802.11 standard [105]. Before transmission, a sender first checks to see if the medium is available using carrier-sensing. A sender determines the channel to be idle when the total energy received is less than the clear-channel assessment threshold. In this case, a sender may begin transmission using the following rule: If the medium

has been idle for longer than a distributed inter-frame spacing time (DIFS) period, transmission can begin immediately. Otherwise, a sender waits for DIFS and then waits for a random backoff interval uniformly chosen between $[0, CW_{\min}]$, where $CW_{\min}$ is the minimum contention window.

Our model strikes balance between realism and simplicity in order to support effective model-driven optimization. We make the following assumptions, which help simplify our model:

$A1$) It assumes pairwise interference, *i.e.*, the interference relationship between two links is independent of activities on other links. Previous works show that pairwise interference is a good approximation in real networks [2, 104]. Hence this assumption is widely used in the literature (*e.g.*, [17, 46, 48, 79, 120]). Moreover, for optimizing the routing of multihop wireless networks, it is often more important to capture the interference relationship among links that are not too far apart. For these links, the pairwise interference relationship is likely to be an even better approximation.

$A2$) It assumes that inherent wireless medium loss (*i.e.*, loss under no interfering traffic) and collision loss are independent, which has been commonly used (*e.g.*, [79, 116]).

$A3$) The inherent wireless medium losses at different nodes are independent, which is experimentally validated in [96, 97, 120].

$A4$) Inter-packet delays from a node follow an exponential distribution, as assumed in [48, 79, 116]. This assumption is only needed for deriving overlapping probabilities between two transmissions.

While some of these assumptions do not always hold (*e.g.*, [135] shows

that loss rates of different wireless links may be correlated for some networks), our evaluation results show that our model-driven optimization yields accurate performance estimates despite such simplifications. With these assumptions, we develop a tractable model with $O(E)$ constraints, where E is the number of links.

3.3.3 Our New Model

We develop a simple interference model for multihop wireless networks to capture the interdependency between broadcast sending rates, loss rates, and throughput. Such interdependency can be captured using $O(E)$ constraints, where E is the total number of edges in the network. These constraints can then be incorporated into the optimization problem as interference constraints [C7] shown in Figure 3.1. We present methods to measure the input parameters of the model in Section 3.5.

Our model consists of two main components: (i) a *sender model* that captures the effects of carrier-sensing on a sender's sending rate, and (ii) a *loss model* that captures both inherent loss (*i.e.*, packet loss under no interference) and the effects of overlapping packet transmissions on the collision loss rates for different links.

3.3.3.1 Broadcast Sender Model

Modeling the effects of carrier sense on traffic rates: We divide time into *variable-length slots* (VLS) for each sender i. A variable-length slot may last for either IEEE 802.11 slot time T_{slot} or the transmission time of a packet followed by a DIFS duration. The former occurs when i senses a clear channel but either has no data to transmit or has data but cannot transmit due to a non-zero backoff counter. The latter occurs when i either transmits a packet or waits for a transmission from another

sender to complete.

Let τ_i be the probability for i to start a new packet transmission in a variable-length slot. Clearly, τ_i depends on (i) how often i has data to send, and (ii) the random backoff interval (*i.e.*, $CW_{\min}$). As derived in [17], when i has saturated traffic demand (*i.e.*, it always has data to transmit), on average i performs one transmission every $CW_{\min}/2+1$ variable-length slots (since there is no exponential backoff for broadcast traffic, we have $CW_{\min}/2$ slots for backoff plus 1 slot for the transmission). Therefore, the transmission probability τ_i is bounded by the following *feasibility constraint*:

$$\tau_i \leq \tau_{\max} \triangleq \frac{1}{CW_{\min}/2+1} \quad (\text{for } \forall i). \tag{3.2}$$

Under the pairwise interference model (*i.e.*, A1), whether sender i carrier-senses (and thus defers to) an ongoing transmission of sender j only depends on nodes i and j and is independent of if other senders are transmitting. Let D_{ij} be this carrier sense probability (*i.e.*, probability for node i to defer to node j when node j is transmitting). For convenience, let $D_{ii} = 1$. Let T_i be sender i's sending rate over all flows ($T_i = \sum_f T(f,i)$), VLS_i be its expected VLS duration, and P_i^{idle} be the idle probability of node i. T_i, VLS_i and τ_i have the following approximate relationship, called the *throughput constraints*:

$$T_i = (EP \times \tau_i)/VLS_i, \tag{3.3}$$

$$\begin{aligned} VLS_i &= T_{\text{slot}} P_i^{\text{idle}} + (T_{\text{xmit}} + T_{\text{DIFS}})(1 - P_i^{\text{idle}}) \\ &= T_{\text{slot}} + (T_{\text{xmit}} + T_{\text{DIFS}} - T_{\text{slot}})(1 - P_i^{\text{idle}}), \end{aligned} \tag{3.4}$$

$$P_i^{\text{idle}} = \prod_j \left(1 - D_{ij} \times \tau_j \times \tfrac{VLS_i}{VLS_j}\right), \tag{3.5}$$

where EP is the expected packet payload size, EH is expected header size, $T_{\text{xmit}} = (EP + EH)/rate$ is the expected packet transmission time, and T_{slot} is an IEEE

802.11 slot time. Eq. (3.3) computes throughput as the total amount of payload transmitted during one VLS divided by the expected VLS duration. Eq. (3.4) computes expected VLS duration as idle probability times an idle slot duration plus transmission (including collision) probability times a transmission duration. Finally, Eq. (3.5) gives the probability that i finds the medium is idle, where $\tau_j \times \frac{VLS_i}{VLS_j}$ is the probability for j to start a transmission in i's VLS, $D_{ij} \times \tau_j \times \frac{VLS_i}{VLS_j}$ is the probability that i defers to j's transmission, and $\prod_j (1 - D_{ij} \times \tau_j \times \frac{VLS_i}{VLS_j})$ is the probability that i does not defer to any node in the network including its own transmission (*i.e.*, i senses the medium is idle).

Eliminating model parameters $\{\tau_i\}$ and $\{P_i^{\text{idle}}\}$: To better facilitate model-driven optimization, we eliminate model parameters $\{\tau_i\}$ and $\{P_i^{\text{idle}}\}$ and transform (3.2)–(3.5) into the following equivalent constraints, which apply directly to the traffic rates $\{T_i\}$.

• *Feasibility constraint.* According to Eq. (3.3), we have: $\tau_i = \frac{T_i \times VLS_i}{EP}$. As a result, Eq. (3.2) is equivalent to:

$$\frac{T_i}{EP} \leq \frac{\tau_{\max}}{VLS_i} \quad (\text{for } \forall i). \tag{3.6}$$

• *Throughput constraint.* With $\tau_i = \frac{T_i \times VLS_i}{EP}$, Eq. (3.5) becomes: $P_i^{\text{idle}} = \prod_j \left(1 - \frac{D_{ij} \times T_j \times VLS_i}{EP}\right)$. So Eq. (3.4) becomes:

$$VLS_i = T_{\text{slot}} + (T_{\text{xmit}} + T_{\text{DIFS}} - T_{\text{slot}}) \times \left[1 - \prod_j \left(1 - \frac{D_{ij} \times T_j \times VLS_i}{EP}\right)\right]. \tag{3.7}$$

Eq. (3.6) and (3.7) fully capture the relationships in (3.2)–(3.5) but have fewer variables. Moreover, note that when traffic rates $\{T_j\}$ are given as inputs, Eq. (3.7) contains only a single variable: VLS_i. This allows us to numerically

derive VLS_i and partial derivatives $\frac{\partial VLS_i}{\partial T_j}$ from the given $\{T_j\}$ (as described in Section 3.4.2). We will therefore use (3.6) and (3.7) in our model-driven optimization.

3.3.3.2 Broadcast Loss Model

Integrating inherent loss and collision loss: To estimate loss rates $P(i,j)$ from traffic rates T_i, we distinguish between two types of loss: inherent wireless medium loss (*i.e.*, loss rate under no interference) and collision loss. The former is denoted as $P^{\text{raw}}(i,j)$ for link $i \rightarrow j$ and can be periodically measured. The latter depends on two factors: (i) how often transmissions from different nodes overlap and (ii) how often such overlapping transmissions result in a collision. To capture the first effect, we introduce $O(i,k)$ to denote the probability for an i's transmission to overlap with a k's transmission (conditioned on i's transmission) and derive its value based on the carrier sense probability. To capture the second effect, we observe that the pairwise interference model indicates there is a constant conditional collision loss probability L_{ij}^k (*i.e.*, the probability that a transmission on link $i \rightarrow j$ collides with an overlapping transmission from node k). We assume that inherent wireless medium loss and collision loss are independent, which has been commonly used (*e.g.*, [79, 116]). We then compute $P(i,j)$ as:

$$P(i,j) = 1 - (1 - P^{\text{raw}}(i,j)) \times \prod_{k \neq i} \left[1 - L_{ij}^k \times O(i,k)\right]. \tag{3.8}$$

This is because a packet is delivered when it is not lost due to either inherent loss or collision loss. To ensure no collision, the packet should not collide with any node's transmission. Since $L_{ij}^k \times O(i,k)$ is the collision loss probability with node k's transmission, $\prod_{k \neq i} \left[1 - L_{ij}^k \times O(i,k)\right]$ is the probability that the link has no collisions with any other node in the network.

Estimating overlap probabilities: We next estimate the overlap probability $O(i,j)$, which depends on whether i and j can carrier sense each other. Our model has two salient features: (i) it supports both symmetric and asymmetric deferral (*e.g.*, node i defers to node j but not vice versa), and (ii) it handles non-binary deferral (*e.g.*, node i sometimes defers to j and sometimes does not).

To provide both features, our modeling strategy is to divide time into regions to which one of the following four cases applies:

- Case 1: i and j can both carrier sense each other;
- Case 2: neither i nor j can carrier sense each other;
- Case 3: i can carrier sense j but j cannot carrier sense i; and
- Case 4: i cannot carrier sense j but j can carrier sense i.

Let $Q_c(i,j)$ be the probability for Case c to occur. Let $O_c(i,j)$ be the probability for a transmission of i to overlap with any transmission of j under Case c. We then have:

$$O(i,j) = \sum_{c=1}^{4}(Q_c(i,j) \times O_c(i,j)). \tag{3.9}$$

Assuming whether i can carrier sense j is independent of whether j can carrier sense i, we can simply compute $Q_c(i,j)$ as:

$$\begin{cases} Q_1(i,j) &= D_{ij} \times D_{ji}, \\ Q_2(i,j) &= (1-D_{ij}) \times (1-D_{ji}), \\ Q_3(i,j) &= D_{ij} \times (1-D_{ji}), \\ Q_4(i,j) &= (1-D_{ij}) \times D_{ji}. \end{cases} \tag{3.10}$$

In our technical report [122], we derive $O_c(i,j)$ as follows.

$$\begin{cases} O_1(i,j) &= \tau_j = \frac{T_j \times VLS_j}{EP}, \\ O_2(i,j) &= 1-(1-\theta_j)\exp\left[-T_{\text{xmit}}/IPD_j\right], \\ O_3(i,j) &= 1-\exp\left[-T_{\text{xmit}}/IPD_j\right], \\ O_4(i,j) &= \frac{\theta_j}{\theta_j+(1-\theta_j)\exp[-T_{\text{xmit}}/IPD_j]}, \end{cases} \tag{3.11}$$

where $\theta_j = \frac{T_j}{rate} \times \frac{EP+EH}{EP}$ is the fraction of time j is transmitting (either payload or header) and $IPD_j \triangleq \frac{1-\theta_j}{\theta_j} \times T_{\text{xmit}}$ is j's expected inter-packet delay.

3.3.3.3 Model Initialization

Our model has the following input parameters: (i) inherent wireless link loss rates P_{ij}^{raw}, (ii) carrier sense probabilities D_{ij}, and (iii) conditional collision loss probabilities L_{ij}^{k}. For simplicity, we estimate these parameters by conducting pairwise broadcast measurements [2, 79], but our model can just as easily use the inputs inferred by more scalable approaches (*e.g.*, [4, 5]).

1. We first let node a send broadcast traffic alone. The other nodes record the receiving rates from a. We then estimate
   $P^{\text{raw}}(a, b) = 1 - (b\text{'s receiving rate from } a)/(a\text{'s sending rate})$.

2. We next let two nodes a and b send broadcast traffic simultaneously and measure their sending rates T_a and T_b. Since neither a nor b has any rate limit, we have $\tau_a = \tau_b = \tau_{\max} = \frac{1}{CW_{\min}/2+1}$. From Eq. (3.3), we can then compute $VLS_a = (EP \times \tau_a)/T_a$ and $VLS_b = (EP \times \tau_b)/T_b$. Applying Eq. (3.7) to the case with only two senders a and b, we have:

$$VLS_a = T_{\text{slot}} + (T_{\text{xmit}} + T_{\text{DIFS}} - T_{\text{slot}}) \times \left[1 - \left(1 - \frac{D_{aa} \times T_a \times VLS_a}{EP}\right)\left(1 - \frac{D_{ab} \times T_b \times VLS_a}{EP}\right)\right]. \quad (3.12)$$

   Note that $D_{aa} = 1$. So linear equation (3.12) has only a single unknown D_{ab}. We can therefore estimate D_{ab} by solving (3.12).

3. Finally, when both a and b are sending broadcast traffic, the other nodes record their receiving rates from a and b. For any node $c \notin \{a, b\}$, we can compute the loss rate $P(a, c) = 1 - (c\text{'s receiving rate from } a)/T_a$. Moreover, given T_a,

```
▷ T: traffic rates, Y: information, P: loss rates
initialization: T* = 0, Y* = 0, thruput* = 0
for k = 1 to KMAX
    P* = estimate_loss(T*)
    [VLS*, ∂VLS*/∂T*] = estimate_VLS_and_partial_derivatives(T*)
    derive linearized interference constraints in Eq. (3.14) using VLS* and ∂VLS*/∂T*
    construct a linear program (LP_k) from Figure 3.1 by adding linearized
    interference constraints (3.14), and fixing loss rates P = P* as constants
    solve (LP_k); let (T^opt, Y^opt) be the optimal solution
    α = α_max; succ = false
    while (α ≥ α_min) and (succ = false) // line search for a better solution
        T = (1 − α) × T* + α × T^opt
        feasible = test_traffic_rates_feasibility(T)
        if (feasible)
            [thruput, Y] = compute_OR_thruput_from_traffic_rates(T)
            if (thruput > thruput*)
                thruput* = thruput; T* = T; Y* = Y;
                succ = true; break
            end
        end
        α = α/2
    end
    if (succ = false), break; end
end
return (thruput*, T*, Y*)
```

Figure 3.2: Iterative optimization of opportunistic routing.

T_b, D_{ab} and D_{ba}, we can compute the overlapping probability $O(a, b)$ according to Eq. (3.9)–(3.11). Applying Eq. (3.8) to the case in which there are only two senders a and b, we obtain:

$$P(a, c) = 1 - (1 - P^{\text{raw}}(a, c)) \times (1 - L^b_{ac} \times O(a, b)). \tag{3.13}$$

We can then estimate L^b_{ac} by solving linear equation (3.13), which has only a single unknown L^b_{ac}.

3.4 Model-Driven Optimization

3.4.1 Iterative Model-driven Optimization

The interference constraints [C7] in Figure 3.1 consist of Eq. (3.6)–(3.11), which capture the inter-dependency between $\{T_i\}$, $\{VLS_i\}$ and $\{P(i, j)\}$. A key challenge in optimization is that these relationships are non-convex. To address

this challenge, we perform optimization in an iterative fashion, as illustrated in Figure 3.2. To decouple the non-linear inter-dependency between $\{T_i\}$, $\{VLS_i\}$, and $\{P(i,j)\}$, we perform the following steps in each iteration:

1. We first fix traffic rates $\{T(f,i)\}$ to their values $\{T^*(f,i)\}$ obtained in the previous iteration and estimate the loss rates $\{P^*(i,j)\}$ as described in Section 3.3.3.2.

2. We then numerically compute VLS_i^* and partial derivatives $\frac{\partial VLS_i^*}{\partial T_k^*}$ from $\{T_j^*\}$ according to Eq. (3.7). The key observation we leverage is that when $\{T_j\}$ are given, Eq. (3.7) only contains a single variable, *i.e.*, VLS_i. We present the details of this step later in Section 3.4.2.

3. We then approximate the non-linear interference constraints given in Eq. (3.6) and (3.7) using linear constraints. This can be achieved by computing the first-order approximation to the R.H.S. of (3.6) as a Taylor expansion at the current T_i^*. Specifically, we use the following linearized interference constraints:

$$\frac{T_i}{EP} \leq \frac{\tau_{\max}}{VLS_i^*} - \frac{\tau_{\max}}{(VLS_i^*)^2} \sum_k \frac{\partial VLS_i^*}{\partial T_k^*} \times (T_k - T_k^*), \tag{3.14}$$

   where VLS_i^* and $\frac{\partial VLS_i^*}{\partial T_k^*}$ are computed in step 2.

4. We then treat loss rates $P^*(i,j)$ as constants in Figure 3.1. We also add the linearized interference constraints given in Eq. (3.14) to the formulation in Figure 3.1, yielding a linear program (LP_k) that can be solved efficiently by LP solvers like `cplex`.

5. Since the linearized interference constraints are only an approximation to the true interference constraints, the optimal solution to (LP$_k$) may be infeasible under IEEE 802.11. We therefore perform a line search between the old solution and the optimal solution to (LP$_k$) to find a new set of traffic rates that

are both feasible and improves the total throughput. During the line search, we need two capabilities: (i) to test whether a set of traffic rates are feasible under 802.11 (line 11 in Figure 3.2), and (ii) to find the maximum total throughput of opportunistic routing under such traffic rates. The former is performed as described in Section 3.4.2. The latter can be achieved by treating $T(f, i)$ as constants while solving the problem formulated in Figure 3.1.

The iterative process continues until it reaches a solution that cannot be further improved upon after enough attempts. Since the total throughput will strictly increase over each iteration, the process is guaranteed to converge. In our experiments, we conservatively limit the maximum number of iterations to 30. Our experience suggests that typically the iteration stops much earlier.

3.4.2 Technical Details

Our model-driven optimization framework above makes use of the following three key capabilities: (i) estimating VLS_i from traffic rates $\{T_j\}$, (ii) testing the feasibility of given traffic rates $\{T_j\}$, and (iii) computing partial derivatives $\frac{\partial VLS_i}{\partial T_k}$. Below we present details on how to support these capabilities using our model.

Estimating VLS_i from traffic rates $\{T_j\}$: To numerically derive VLS_i from given traffic rates $\{T_j\}$, let $f_i(x) \triangleq x - T_{\text{slot}} - (T_{\text{xmit}} + T_{\text{DIFS}} - T_{\text{slot}}) \times \left[1 - \prod_j \left(1 - \frac{D_{ij} \times T_j \times x}{EP}\right)\right]$. According to Eq. (3.7), $x = VLS_i$ is a root of $f_i(x)$. Moreover, we need $x \in \left[0, \frac{EP}{\max_j (D_{ij} \times T_j)}\right]$ to ensure $1 - \frac{D_{ij} \times T_j \times x}{EP} \geq 0$ in Eq. (3.7). In our technical report [122], we prove that when $x \in \left[0, \frac{EP}{\max_j (D_{ij} \times T_j)}\right]$, $f_i(x)$ is convex and has at most one root. Therefore, we can apply any univariate root-finding algorithm (*e.g.*, Matlab's fzero function) to numerically compute the root of $f_i(x)$ over interval $x \in \left[0, \frac{EP}{\max_j (D_{ij} \times T_j)}\right]$ and let the solution be VLS_i (if a root exists).

Testing the feasibility of traffic rates $\{T_j\}$: To test whether traffic rates $\{T_j\}$ are feasible, we first numerically compute VLS_i from Eq. (3.7) by finding a root of $f_i(x)$ over $x \in \left[0, \frac{EP}{\max_j(D_{ij} \times T_j)}\right]$ as described above. If no solution is found or if the solution VLS_i violates Eq. (3.6), then traffic rates $\{T_j\}$ are infeasible. Otherwise, $\{T_j\}$ are feasible.

Computing partial derivatives $\frac{\partial VLS_i}{\partial T_k}$: Eq. (3.7) also allows us to compute the partial derivatives $\frac{\partial VLS_i}{\partial T_k}$ for given traffic rates $\{T_j\}$, which allows us to linearize the non-linear interference constraints (see Section 3.4.1). Specifically, we have $\frac{\partial VLS_i}{\partial T_k} = \frac{N_{ik}}{1-M_i}$, where $M_i \triangleq (T_{\text{xmit}} + T_{\text{DIFS}} - T_{\text{slot}}) \times P_i^{\text{idle}*} \times \sum_j \frac{D_{ij}T_j}{EP - D_{ij}T_j VLS_i}$, $N_{ik} \triangleq (T_{\text{xmit}} + T_{\text{DIFS}} - T_{\text{slot}}) \times P_i^{\text{idle}*} \times \frac{D_{ik}VLS_i}{EP - D_{ik}T_k VLS_i}$, and $P_i^{\text{idle}*} \triangleq \prod_j (1 - \frac{D_{ij}T_j VLS_i}{EP})$.

3.5 Protocol Implementation

Overview: We develop a practical opportunistic routing protocol to install the opportunistic routes and rate limits computed by our optimization algorithm. It is built on top of MORE [27], which sits between the IP and 802.11 MAC layers. It differs from MORE in that it uses interference modeling and optimization to derive rate limits and opportunistic routes for a given performance objective. As in MORE, it leverages intra-flow network coding to carry out the derived routes (*i.e.*, an intermediate forwarder transmits random linear combinations of the packets it receives for a given flow at the rate derived from our optimization).

As most opportunistic routing protocols, we target medium to large file transfers. A traffic source divides data packets into batches, and broadcasts a random linear combination of the original packets at the rate computed according to Figure 3.2. Upon receiving encoded packets, an intermediate node generates a ran-

dom linear combination of all the innovative packets it has from the current batch. Each intermediate node uses the algorithm described in Figure 3.2 to determine how much traffic it should forward. After receiving enough innovative packets, the destination extracts the original data packets and sends an end-to-end ACK using MAC-layer unicast. When the source receives the ACK, it moves to the next batch. Below we describe several key steps in our protocol: (i) measuring inputs to seed our interference model, (ii) computing opportunistic routes and rate limits for each flow, (iii) routing traffic according to the derived sending rates and routes, (iv) supporting multicast, and (v) enhancing the reliability of end-to-end ACKs.

Measuring input parameters: Our model-driven optimization framework has the following input parameters: (i) traffic demands, (ii) carrier sense probabilities, (iii) conditional collision loss probabilities, and (iv) inherent wireless link loss rates. As reported in [44, 79], wireless traffic exhibits temporal stability and we can estimate current traffic demands based on previous demands. In our evaluation, we also test the sensitivity to the demand estimation error. We conduct pairwise broadcast measurements [2] and compute the carrier sense probabilities $\{D_{ab}\}$ and conditional collision loss probabilities $\{L^b_{ac}\}$ as described in Section 3.3.3.3. The pairwise broadcast measurements takes $O(N^2)$ time for an N-node network. In our 21-node testbed, each pair of nodes broadcasts for 30 seconds, and the entire measurement takes around 2 hours. To minimize measurement overhead, we conduct pairwise broadcast measurement infrequently, around once a week. Note that recent works have developed efficient online techniques to measure interference when a network is in use (*e.g.*, [4, 5]). These techniques can be incorporated into our implementation to further reduce measurement overhead. In addition, we conduct per-node broadcast measurements at the beginning of each experiment to measure the inherent wireless link loss rates. The latter is based on more frequent measurements

because it is more light-weight (only requiring $O(N)$ measurements) and existing routing protocols, such as [18, 27, 31], all use frequent loss measurements for route selection.

Deriving opportunistic routes and rate limits: Note that since our optimization problem is non-convex, existing techniques developed for distributed convex optimization (*e.g.*, [68]) are not directly applicable. Instead, we optimize opportunistic routes and rate limits at a central location and then distribute the optimized results to the other nodes. We use this approach in our implementation. The amount of information to distribute is very small compared to data traffic: the optimization input is around 2 KB per node and the optimization output is within a 100 bytes per node. Alternatively, the computation can also be done in a fully distributed fashion, similar to link-state protocols like OSPF, where every node implements the same algorithm over the same data to arrive at the same results. Such computation happens once every several minutes. For instance, default SNMP polling intervals are typically 5 minutes, so the optimization can rerun when the traffic demands and network topology change. The optimization is fairly efficient (*e.g.*, it takes around 3 seconds to optimize routes and rate limits for 16 flows under the 5×5 grid topologies used in our simulation).

Enforcing derived routes and rate limits: An intermediate node enforces its forwarding strategy according to the derived $T(f, i)$ and $Y(f, d, i, j)$ using the following credit-based scheme. When node j receives a packet from node i, it increments its credit, which denotes how many packets j should transmit for each received packet. If its credit is at least 1, j generates and transmits a random linear combination of the packets from the current batch buffered locally, and decrements the credit by 1. This process repeats until the credit goes below 1. The credit computation in our protocol differs from MORE in two main aspects. First, our protocol

computes credit to ensure the traffic and information sending rates conform to the derived T and Y. Second, unlike MORE, which treats all transmissions equally if coming from nodes with larger ETX to the destination, our protocol differentiates transmissions coming from different neighbors as follows. Upon receiving a packet from i, j increments its credit by $C \times R$, where C reflects the fraction of useful information contained in each packet received from i and R reflects the amount of redundancy j should include to compensate for loss to its neighbors. Specifically, we have $C = \frac{Y(f,d,i,j)}{T(f,i)(1-P(i,j))}$, and $R = \frac{T(f,j)}{\sum_k Y(f,d,j,k)}$. For example, when j receives a packet from a downstream node i, $C = 0$ to prevent j from sending non-innovative packets; when receiving a packet from an upstream node i, j updates its credit according to how much new information is involved in the packet and its loss rate to its forwarders.

Supporting multicast extension: Our previous description applies to the unicast case. A few modifications are required to support multicast. First, since a single packet carries a different amount of information for different destinations in the same multicast group, a node j increments its credit by $C \times R$, where $C = \frac{\max_d Y(f,d,i,j)}{T(f,i)(1-P(i,j))}$ and $R = \frac{T(f,j)}{\sum_k \max_d Y(f,d,j,k)}$. Second, when some destinations receive enough innovative packets, the encoded packets from the current batch should only be delivered to those who have not received all packets. To adapt to the changes in the set of destinations that need the packets, we dynamically re-adjust credit increment based on the remaining receivers who have not finished.

Enhancing ACK reliability: The destination sends an end-to-end ACK to the source upon receiving enough innovative packets for decoding so that the source can move on to the next batch. To ensure the reliability of ACKs, we keep retransmitting ACKs until they are received. To expedite ACK transmissions, ACKs do not perform binary backoff so that they have higher priority over retransmit-

ted data. For fair comparison, we apply the same optimizations to MORE. Finally, since there is only one ACK for a batch of data packets, ACK overhead is negligible in opportunistic routing.

3.6 Evaluation of Accurate Model-Driven Optimization Framework

3.7 Evaluation Methodology

We evaluate our approach using extensive simulation and testbed experiments. Our evaluation consists of four parts. First, we compare the fidelity of the conflict-graph (CG) model and our new model by quantifying their under-prediction and over-prediction errors. We use a conservative CG model, which considers two broadcast transmissions to interfere if any one of their receivers is interfered by the other transmission.

Second, we compare the performance of our opportunistic routing protocol using either the CG model or the new broadcast interference model with the following existing routing protocols: (1) shortest-path routing using the ETX routing metric, which minimizes the total number of expected transmissions from a source to its destination [31], (2) shortest-path routing with rate limit optimization as developed in [79], and (3) MORE, a state-of-art opportunistic routing protocol.

We compare total network throughput under 1–16 simultaneous flows. We also compare in terms of the proportional fairness metric [68], which is defined as: $\sum_{f \in Flows} \log G(f)$, where $G(f)$ is flow f's throughput. This metric strikes a balance between increasing network throughput and maintaining fairness among the flows. Higher values are more desirable. Unless otherwise noted, each flow sends saturated CBR traffic.

Third, we evaluate the multicast performance of one multicast group with a varying group size, and measure the average throughput of the bottleneck receiver. As in [27], we extend shortest path routing to support multicast by generating a multicast tree as a union of shortest paths towards all destinations and sending one copy of traffic along the links that are shared by multiple destinations. It saves the traffic on shared point-to-point links as in wire-line multicast routing but does not leverage the broadcast nature of wireless links (*e.g.*, a node still needs to send traffic separately to reach each of its next hops). Shortest path with rate limit [79] takes a routing matrix R as part of the input, where R_{id} is the fraction of flow d that traverses link i. To support multicast, we derive a multicast routing tree R, where $R_{ig} = 1$ if link i appears in multicast group g's routing tree.

Fourth, we evaluate the sensitivity of our protocol against (i) errors in the input traffic demands, (ii) unknown external interference, and (iii) errors in link loss estimation.

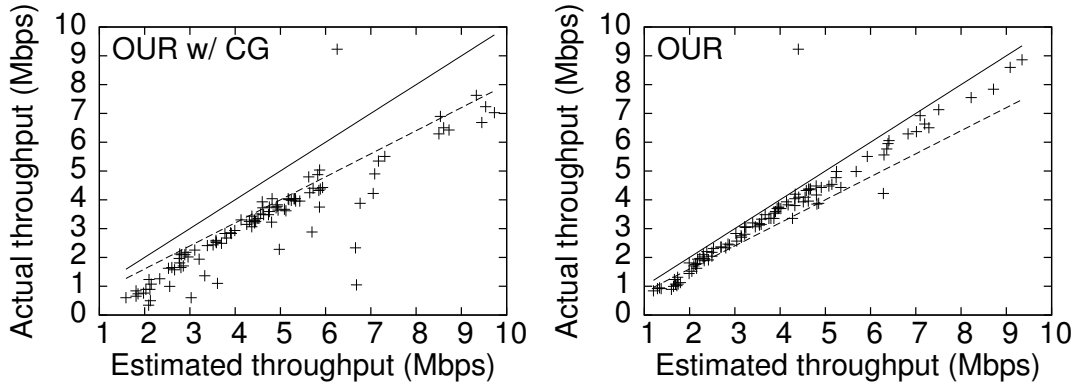
For simulation, we implement all protocols in Qualnet 3.9.5 [117]. For testbed experiments, we use the shortest path routing and MORE implementations publicly available [99]. In particular, the shortest path routing is the Click implementation released as part of MORE source code. We calculate ETX according to [31] and configure the link weight accordingly. The shortest path with rate limiting is based on the shortest path code but the rate limit of each flow is computed using the algorithm in [79]. We extend MORE to implement our protocol as described in Section 3.5. Both MORE and our protocol use 64 packets as the batch size for network coding. All these routing protocols are implemented using Click [30] and the MadWiFi driver [90] in the testbed.

Qualnet simulation: In simulation, we use 802.11a with a fixed MAC rate of 6 Mbps. The communication range is 230 meters, and interference range is 253 me-

ters. These are the default values in Qualnet under transmission power of 10dBm, and we use them in the CG model to determine if two nodes interfere. As in [79], we seed the new interference model by having two senders broadcast simultaneously and measuring the resulting sending rates and receiving rates. Unless noted otherwise, we use saturated UDP traffic with 1024-byte payloads.

For each scenario, we conduct 20 random trials. In each trial, flow sources and destinations are picked randomly and the simulation time is 20 seconds. We extend Qualnet to generate directional inherent packet losses, which are uniformly distributed between 0 and 90%. We consider two types of topologies: 5×5 grid and 25-node random topologies, each occupying a $750 \times 750\ m^2$ area.

Testbed experiments: Our testbed consists of 21 nodes located on two floors inside an office building. Each node runs Linux and is equipped with a NetGear WAG511 NIC. Unless otherwise specified, we use 802.11a to minimize interference with campus wireless LAN traffic, which uses 802.11g. This allows us to evaluate in a controlled environment. We use 20 mW transmission power and 6 Mbps transmission rate so that the network paths are up to 7 hops. Among the node pairs that have connectivity, 47.8% of them have links with loss $\leq 20\%$. All the routing protocols require estimation of link loss rates, which are measured by having one sender broadcast at a time and the other nodes measure the receiving rates. The loss measurements were collected before the experiments. In addition, our protocol and shortest path with rate limiting require interference measurement, which we collected once per week. As in simulation, we conduct 20 random trials for each scenario. Each trial lasts one minute. Other settings are consistent with the simulation. Finally, in Section 3.10, we further evaluate using 802.11b, which competes with campus WLAN traffic, in order to assess the sensitivity against unknown external interference.



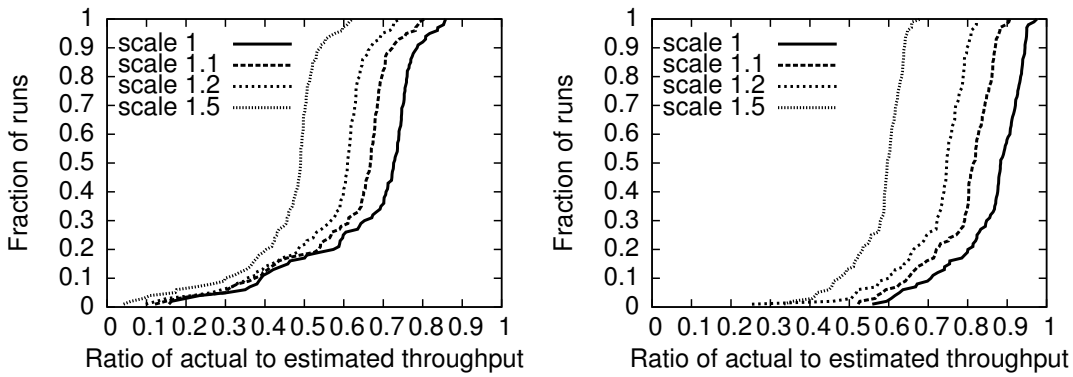
(a) CG model (b) Our model

Figure 3.3: Actual vs. estimated throughput in simulation (25-node random topologies).

3.8 Model Validation

We adopt the evaluation methodology presented in [79] to quantify the accuracy of our model. In particular, to evaluate the over-prediction of our model, we install the estimated throughput to the network to see if it can be satisfied. To evaluate the under-prediction error, we uniformly scale each flow throughput by the same factor and check if the scaled demand is achievable. If the scaled demand is achieved in the network, it indicates that the under-prediction error is at least the scaling factor. We vary the scaling factor from 1.1, 1.2, 1.5, corresponding to a load increase of 10%, 20%, and 50%, and vary the number of flows from 1 to 16.

Simulation results: We first evaluate how often the models over-predict. In Figure 3.3, we plot the estimated throughput versus the actual throughput using the CG model and our model in 25-node random topologies. For reference, we plot lines $y = x$ and $y = 0.8x$. Here, the CG model significantly over-predicts the actual throughput obtained, whereas the actual performance under our model is mostly within 80% of the estimated throughput. The CG model experiences significantly higher over-prediction errors since it assumes perfect scheduling, whereas

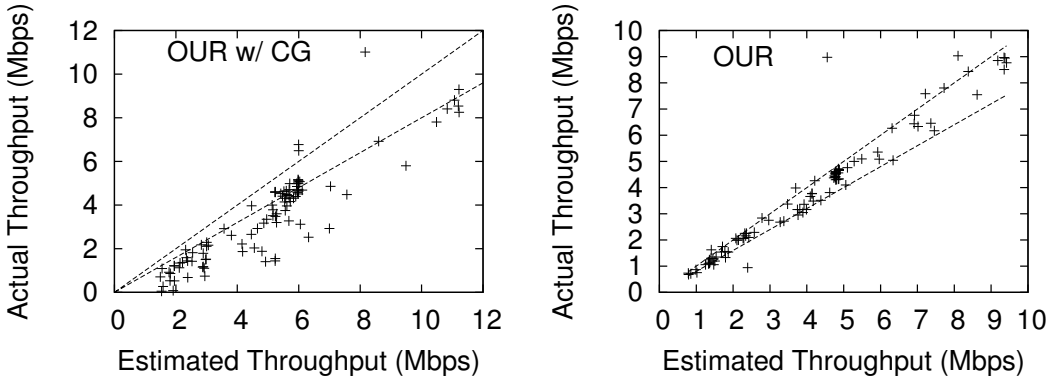


(a) CG model (b) Our model

Figure 3.4: CDF of ratios between actual and estimated throughput in simulation (25-node random topologies).

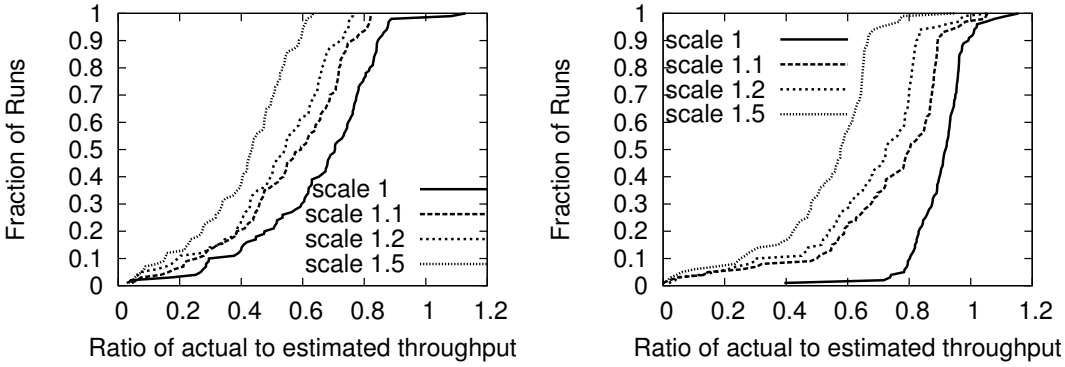
our model explicitly models the interference between broadcast transmissions in IEEE 802.11, thereby achieving higher accuracy. Moreover, the amount of over-estimation by CG heavily depends on the network topology (*e.g.*, whether the network has hidden terminals) and simply scaling down the performance estimated by CG by a constant factor does not work. Both CG and our models have part of their over-prediction errors coming from the delay in end-to-end ACKs, during which time the source keeps retransmitting the current batch. This effect is not modeled. The use of a larger batch size can reduce the gap between the model estimation and actual performance at the cost of a larger header size and longer delay.

Next we quantify under-prediction errors. In Figures 3.4(a) and (b), we plot CDFs of the ratios between actual and estimated throughput in random topologies for the CG model and our model, respectively. Consistent with the scatter plots, the CG model mostly over-predicts, and virtually none of the scaled demands are satisfied. In comparison, using our model with a scale factor of 1, 80% of the runs have actual throughput within 80% accuracy of the estimation. Increasing the scale factor to 1.1 causes 65% of the actual throughput to be within 80% accuracy. After



(a) CG model (b) Our model

Figure 3.5: Actual vs. estimated throughput in the testbed.



(a) CG model (b) Our model

Figure 3.6: CDF of ratios between actual and estimated throughput in the testbed.

a further increase of the scale factor to 1.2, only 11% of actual throughput falls into 80% accuracy. This indicates that the demands scaled up by 20% can rarely be satisfied and shows our model has low under-prediction errors.

Testbed results: Next we validate our model and the CG model using testbed experiments. Figures 3.5(a) and (b) show the scatter plots of the CG model and our model, respectively. Figures 3.6(a) and (b) plot the CDFs of the ratios between actual and estimated throughput using different scale factors. As in simulation, the scatter plots from testbed experiments show a good match between actual and

estimated throughput using our model and a significant over-estimation in the CG model. Scaling the demands by 1.1 leads to only 50% of the demands being satisfied and scaling the demands by 1.2 leads to only 29% of the demands being satisfied. These results indicate low over-prediction and under-prediction error. There are a few points in the testbed results where the actual throughput is higher than the estimated throughput. These cases arise from loss fluctuation: we use loss measurements to seed our model and derive opportunistic routes and rate limits, but the actual link loss rates in the experiment improve and support higher throughput.

Summary: The simulation and testbed results demonstrate that our model rarely over-estimates or under-estimates performance by more than 20%. In comparison, the CG model consistently over-predicts network throughput due to its assumption of perfect scheduling. These results highlight the importance of model fidelity on performance predictability.

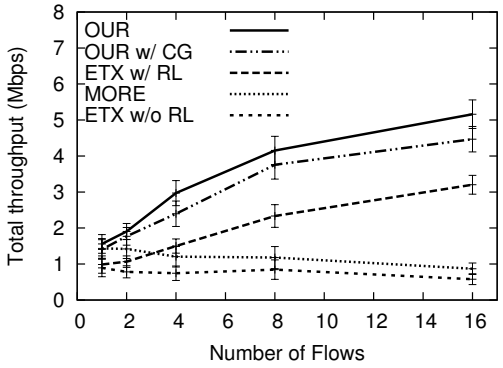
3.9 Performance Comparison

In this section, we compare the performance of different routing protocols using simulation and testbed experiments.

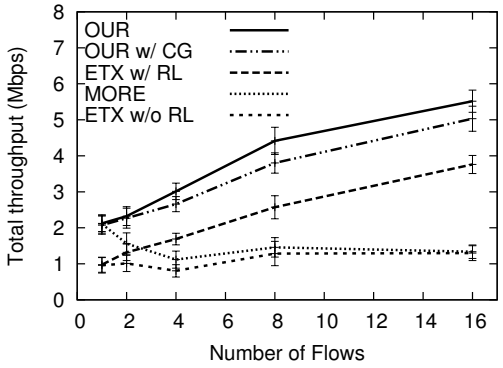
3.9.1 Simulation Results

Total throughput of unicast flows: Figures 3.7(a) and (b) show the total throughput for 5×5 grid and 25-node random topologies, respectively. The error bars on the graph show the standard deviation of the sample mean.

We make several observations. First, in all cases our protocol using our model yields the best performance. It out-performs ETX by 76%-799% in the grid topology and by 117%-327% in the random topologies. Its gain over ETX with rate



(a) 5×5 grid



(b) 25-node random topology

Figure 3.7: Total unicast throughput in simulation (25-node random topologies).

limiting ranges from 57%-99% in the grid topology and 46%-117% in the random topology. Its gain over MORE increases rapidly with the number of flows, ranging from 34% (2 flow) to 146% (4 flows) to 501% (16 flows) in the grid topology, and from 50% (2 flows) to 169% (4 flows) to 311% (16 flows) in random topologies. It out-performs the protocol with CG, the second best performing protocol by up to 24% in the grid topologies and 16% in the random topologies. Its performance benefit comes from three main factors: (i) taking advantage of opportunistic transmissions to cope with lossy wireless links, (ii) using interference-aware rate limiting to avoid network congestion, and (iii) using interference-aware opportunistic routing to maximize spatial reuse.

Second, comparing MORE against shortest path rate limiting, we observe that MORE out-performs the latter under 1 or 2 flows by leveraging opportunistic transmissions to recover losses. As the number of flows increases, the performance of MORE degrades and becomes significantly worse than shortest path with rate limiting due to lack of rate limiting. The impact of rate limiting on opportunistic routing is even higher than shortest path routing because opportunistic routing uses broadcast transmissions, which do not have binary backoff and are more likely to cause network congestion. Further, congestion on the data path may corrupt end-to-end ACKs in opportunistic routing and lead to unnecessary retransmissions and throughput degradation. In contrast, shortest path routing uses unicast transmissions, whose MAC-layer ACKs are given higher priority and hence more reliable.

Multicast flows: Figure 3.8 shows the throughput of the bottleneck receiver in a multicast group as we vary the group size from 1 to 5. As in unicast flows, our protocol consistently out-performs the alternatives. It improves the protocol with CG by 10%-46%, MORE by 8%-47%, shortest path rate limiting by 58%-232%, and shortest path by 74%-894%. The larger performance gain over both versions of

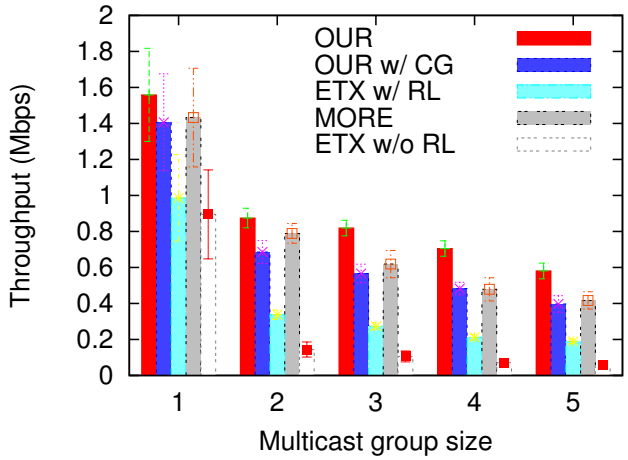


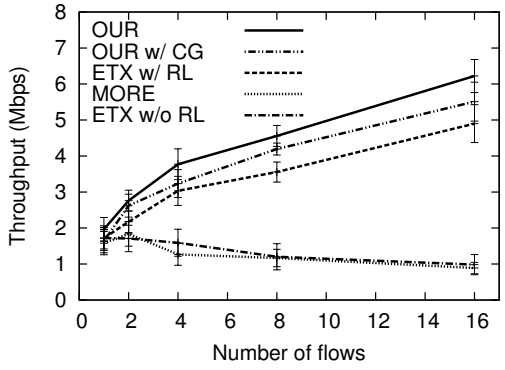
Figure 3.8: Multicast throughput in a 5×5 grid.

shortest path is because our protocol effectively exploits the broadcast nature of the wireless medium to reduce the number of transmissions. When sending to multiple neighbors, it uses one broadcast transmission to reach all the neighbors. In comparison, while shortest path routing uses a multicast tree to compress the traffic on a shared link, the links from one sender to different neighbors are considered different and multiple transmissions are required to reach them. For the same reason, MORE consistently out-performs both versions of shortest path routing. Our protocol still out-performs MORE and the protocol with CG by using a more accurate model to optimize rate limit and opportunistic routes.

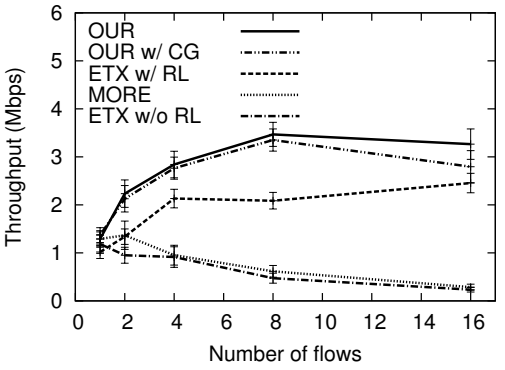
3.9.2 Testbed Results

Throughput of unicast flows: Figure 3.9(a) shows the total throughput of different protocols in the testbed, which has up to 7 hops. The relative rankings of the routing schemes are consistent with the simulation. As before, our protocol yields the best performance. The links in our testbed tend to be binary: either low loss or close to no connectivity. Among the node pairs that have network connectivity, 47.8% of them have loss rate within 20%. So the benefit of opportunistic routing is smaller in the testbed than in simulation. MORE performs close to shortest path routing, and significantly worse than shortest path with rate limiting; similarly, the gap between our protocol and shortest path routing also becomes smaller. These results confirm the intuition that opportunistic routing is most useful under lossy wireless medium.

To understand how opportunistic routing performs under more lossy wireless medium, we conduct another set of experiments where we pick only flows whose ETX between source and destination is at least 1.25. Figure 3.9(b) summarizes the results. In this case, the throughput of our protocol is 1.09-14.0x that of shortest path without rate limiting and 1.26-1.67x that of shortest path with rate



(a) Random flow selection



(b) Flows with ETX > 1.25

Figure 3.9: Total unicast throughput in the testbed.

limiting. Its throughput is similar to MORE under 1 flow and 11.47x MORE's throughput under 16 flows. Furthermore, MORE yields low throughput: its performance is worse than shortest path with rate limiting as the number of flows reaches 4 or higher. These results are consistent with the simulation, and highlight the importance of jointly optimizing rate limits and opportunistic routes.

Proportional fairness of unicast flows: Next we consider maximizing proportional fairness. Since this objective is non-linear, in order to optimize it, we first approximate it using a piecewise linear, increasing, convex function as follows. We select s points on $\log(x)$, and approximate $\log(x)$ using s line segments, each connecting two adjacent points. We perform two different point selections and observe

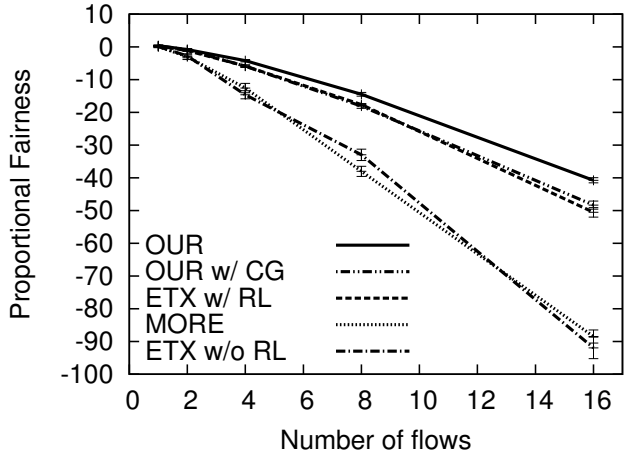


Figure 3.10: Unicast proportional fairness in the testbed.

similar performance. In the interest of space, below we present results from only one selection: $x = 0.001, 0.01, 0.1, \sqrt{0.1}, 1, \sqrt{10}, 10$. When a flow's throughput is 0, its log value is undefined, so we set its throughput to 1 Kbps. Figure 3.10 shows the proportional fairness as we vary the number of unicast flows in the testbed. The three routing schemes that support rate limiting significantly out-perform MORE and shortest path without rate limiting since the latter two can easily cause starvation. Among those that support rate limit, our protocol performs the best due to its opportunistic routing and high-fidelity model.

Multicast flows: Finally, we evaluate the performance of multicast in our testbed. Figure 3.11 shows the throughput of the bottleneck multicast receiver in one multicast group, where the multicast group size is varied from 2 to 4. Our protocol performs the best. It out-performs the protocol with CG by 16%-38%, MORE by 10%-63%, shortest path with rate limiting by 68%-89%, and shortest path routing by 101%-181%. In addition, by leveraging the broadcast wireless medium, all types of opportunistic routing, including MORE, out-perform both versions of shortest path routing. These results suggest opportunistic routing is even more use-

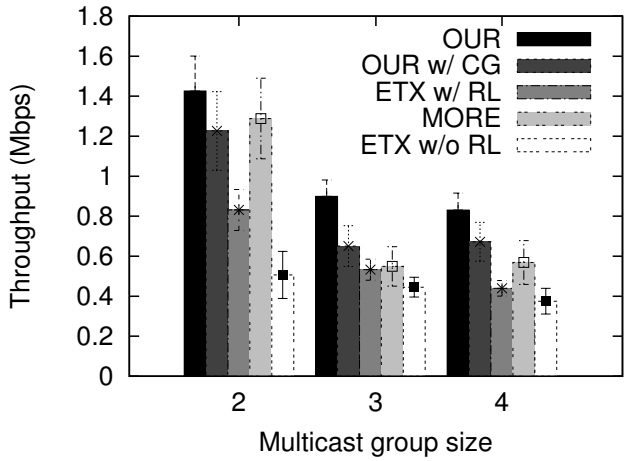


Figure 3.11: Multicast throughput in the testbed.

ful to multicast, and the effective optimization of multicast routes and rate limiting continues to be important.

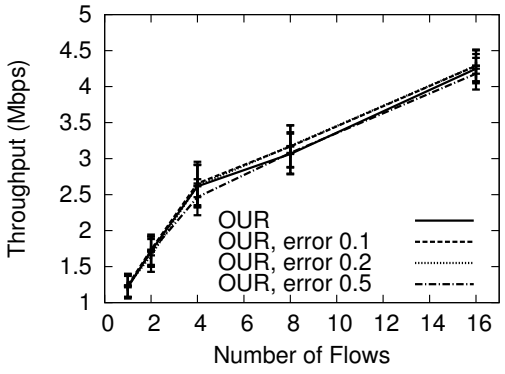
3.9.3 Summary of Performance

The simulation and testbed results show that our protocol consistently outperforms the alternatives. By leveraging opportunistic transmissions and effective route optimization, it significantly out-performs state-of-the-art shortest path routing protocols. By using a high fidelity network model to jointly optimize rate limits and opportunistic routes, it significantly out-performs state-of-the-art opportunistic routing protocols. These benefits suggest that all the design components in our protocol, including opportunistic routing, network model, and joint rate limit and route optimization, are essential and help improve the performance.

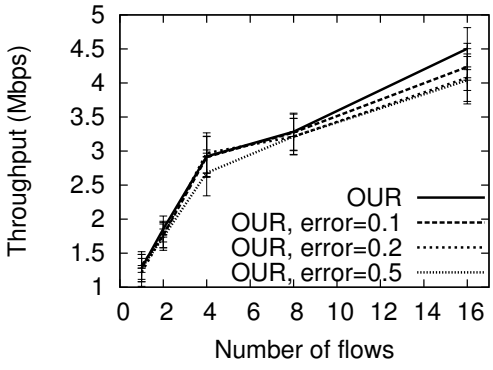
3.10 Evaluation of Sensitivity

3.10.1 Impact of Inaccurate Traffic Demand

Methodology: We first evaluate the performance under inaccurate traffic demand estimation, since in practice traffic demands fluctuate and may not be known exactly. The actual traffic demands are uniformly distributed between 0 and the maximum link throughput. To simulate demand estimation error, we inject errors into the actual demands and feed the salted demands to our optimization framework while imposing the actual demands to the network for evaluation. The error injected is uniformly distributed between 0-10%, 0-20%, and 0-50%. To protect against esti-



(a) 5×5 grid in simulation



(b) 802.11a Testbed

Figure 3.12: Throughput under inaccurate traffic estimates.

mation error, our protocol slightly over-provisions by scaling the derived sending rates from the optimization output by a factor of 1.1.

Simulation: Figure 3.12(a) shows the total throughput versus the number of flows. We see similar performance across different error ranges. This indicates that our protocol is fairly robust against demand estimation errors, because for the purpose of performance optimization, the spatial traffic demand distribution is more important than the exact demand values.

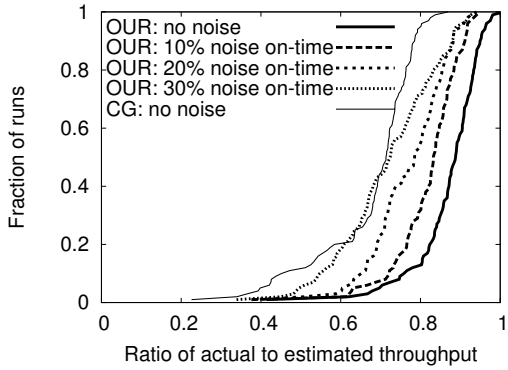
Testbed: Figure 3.12(b) shows the performance of our protocol when we feed inaccurate traffic demands as input to our optimization. As in simulation, it is robust to the inaccuracy in traffic demand estimation in testbed. Its performance under no error is close to that under the relative error of 0.5.

3.10.2 Impact of Unknown External Interference and Loss Fluctuation

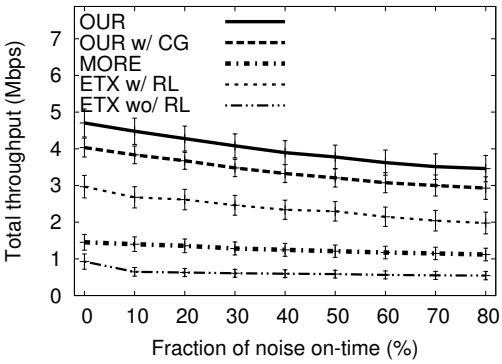
3.10.2.1 Simulation

Methodology: We create external interference by randomly placing two external noise sources in 25-node random topologies. All protocols compute routes and rate limits without considering the external noise, and we measure the throughput of using the derived routes and rate limits under external noise. The noise sources have uniformly distributed on and off time, where the average on-time is 0.25 second and the total simulation time is 20 seconds. We vary the average off-time so that every noise source is on 10% to 80% of time. During on-time, each noise source broadcasts 802.11 packets (with 1024-byte payload) as fast as possible.

Model validation: First, we compare actual throughput under external noise versus estimated throughput derived without considering the noise sources. As shown in Figure 3.13(a), the accuracy of our protocol degrades gracefully as we increase the



(a) CDF of ratios between actual and estimated throughput



(b) Total throughput of 8 flows

Figure 3.13: Simulation results under 2 noise sources with varying on-time in 25-node 802.11a random topologies.

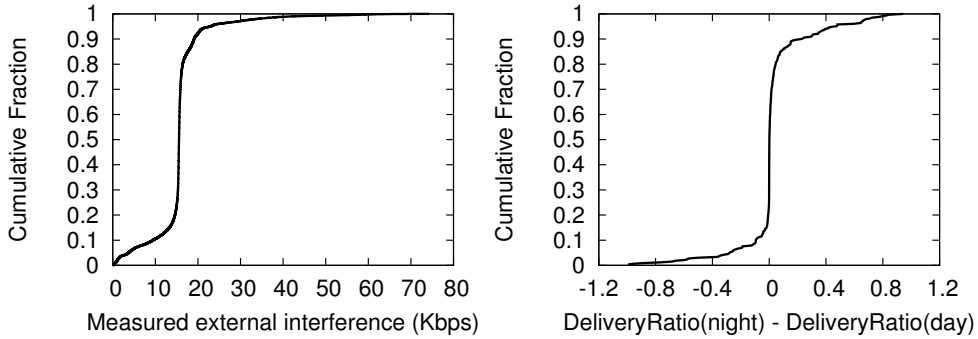
on-time of each noise source. The fractions of runs that achieve within 30% error are 99% under 10% noise on-time, 76% under 20% noise on-time, and 56% under 30% noise on-time. Moreover, even with 30% noise on-time, it achieves much higher predictability than the protocol with CG model under no external noise.

Performance comparison: As shown in Figure 3.13(b), the ranking of different protocols remains the same across all noise levels. Our protocol consistently out-performs all other protocols. Even when every noise source is active 80% of time, it out-performs the one with CG by 18%, shortest path with rate limiting by 75%,

MORE by 209%, and shortest path without rate limiting by 535%. Moreover, the performance of different protocols degrades smoothly as the on-time of each noise source increases.

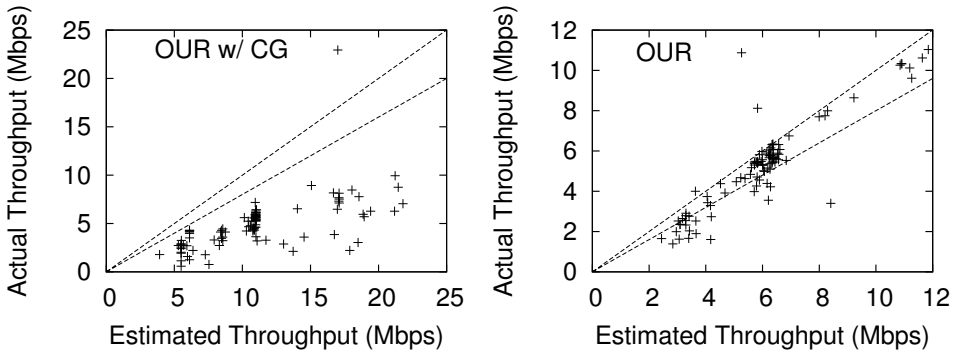
3.10.2.2 Testbed

Methodology: We also evaluate the sensitivity in an 802.11b testbed consisting of 22 nodes. As before, we randomly select flows in our network. As common practice, we run the link loss measurements at night, which has low network activity. Then we run all evaluation during the day. This allows us to evaluate the sensitivity against unknown external interference and loss fluctuation. In particular, our building has an active 802.11g campus network, whose traffic directly interferes with our wireless mesh traffic. We treat traffic from the campus network as unknown external interference. Figure 3.14(a) plots the CDF of the average campus network traffic measured by all mesh nodes in promiscuous mode every 30 seconds. The median and mean are both 15.5 Kbps. Moreover, loss fluctuates from nights to daytime. Figure 3.14(b) plots a CDF of $DeliveryRatio(night) - DeliveryRatio(day)$ over all links that have $\geq 5\%$ delivery rates. We observe loss fluctuation, because



(a) Amount of external traffic (b) CDF of loss fluctuation

Figure 3.14: Amount of external traffic from the campus network and loss fluctuation in our 802.11b testbed.



(a) CG model (b) Our model

Figure 3.15: Actual vs. estimated throughput in 802.11b testbed under unknown external interference and loss fluctuation.

during the day time (i) more people sit near mesh nodes and cause more attenuation, and (ii) more people move around and close/open doors and cause frequent changes to the RF environment.

Model validation: Figure 3.15 shows the scatter plot of actual versus estimated throughput from the 802.11b testbed. We also plot $y = x$ and $y = 0.8x$ for reference. Our protocol continues to exhibit high predictability: 78% of runs have

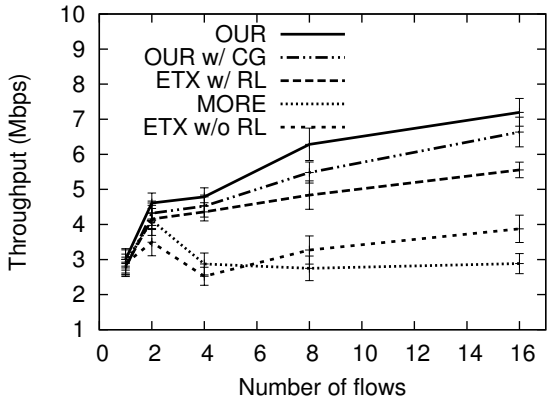


Figure 3.16: Unicast throughput in our 802.11b testbed under unknown external interference and loss fluctuation.

within 20% error.

Performance comparison: As shown in Figure 3.16, our protocol continues to perform the best. Different from simulation, MORE sometimes performs worse than shortest path without rate limiting because the network congestion in MORE is more severe in a dense network like our 802.11b testbed.

3.11 Summary

In this chapter, we present the first protocol that can accurately optimize the performance of opportunistic routing in IEEE 802.11 networks. Our framework consists of three key components: (i) a simple yet effective wireless network model to support optimization, (ii) a novel algorithm for optimizing different performance objectives, and (iii) an opportunistic routing protocol that effectively maps solutions resulted from our optimization into practical routing configurations. Through testbed implementation and simulation, we show that the performance of our protocol is close to our estimation, and is much better than state-of-the-art shortest path routing and opportunistic routing protocols. Moreover, it is robust against inaccuracy introduced by a dynamic network and it also consistently out-performs the existing schemes. To further enhance the robustness against traffic and topology variations, in the future we plan to extend the robust traffic engineering techniques developed in the Internet to optimize wireless networks. In particular, a traffic engineering system usually collects a set of traffic matrices and uses their convex combination to cover the space of common traffic patterns for optimization. These new demand constraints are compact and can be easily incorporated into our framework. We plan to extend this technique to cope with both traffic and topology variations in wireless networks.

Chapter 4

O3: Optimization of Overlay-based Opportunistic Routing

4.1 Introduction

In wireless mesh networks, providing efficient and reliable communication is important because wireless losses are common. In the previous chapter, we observe that opportunistic routing effectively combats wireless losses by taking advantage of the broadcast nature of the wireless medium (*e.g.*, [18, 27, 87]). In this chapter, we explore whether opportunistic routing can reap benefit from inter-flow network coding, which has been successfully applied to single path routing in wireless mesh networks (*e.g.*, COPE [67]).

Motivation: First, let's revisit the motivating example of 4-node diamond topology in Figure 1.1 in Chapter 1 with two bi-directional flows between A and D.

In *traditional single path routing*, the expected number of transmissions to deliver one packet over each hop is 2 due to the 50% loss rates, and altogether 8 transmissions are required to deliver one packet for each of the two flows.

In *opportunistic routing*, a flow source uses either B or C to forward traffic (instead of only B or C). Therefore a packet makes progress if it reaches either forwarding node. This probability is 75%, assuming independent link loss, which is common in many real networks [96, 97, 120]. So on average it takes only 1.33 transmissions to move a packet over the first hop (*i.e.*, to either of the intermediate

nodes) and 2 transmissions to move the packet from the intermediate node to the destination. Therefore, altogether 6.66 transmissions are required to successfully deliver both packets.

The performance of *inter-flow coding* [67] depends on whether there is an inter-flow coding opportunity. If the two flows use the same intermediate node as the forwarder, which is the best case, then it takes 6 transmissions to successfully deliver both packets (*i.e.*, 2 transmissions to deliver one packet over the first hop in both flows as in single path routing, and 2 transmissions for the intermediate node to deliver the packets to both A and D by XOR-ing them). In this case, node A can extract the packet it needs by XOR-ing its own packet with the one received from the forwarder. So can node D. When the two flows use different forwarders, there is no inter-flow coding opportunity and it takes 8 transmissions to deliver one packet for each flow as in the traditional single-path routing.

We propose to exploit inter-flow network coding in opportunistic routing. Not only does it take only 1.33 transmissions to move a packet across the first hop by using opportunistic routing, but also an intermediate node can XOR packets from the two flows whenever possible. In the *best case* (*i.e.*, the intermediate nodes can XOR all packets), the intermediate nodes only need 2 transmissions to deliver packets for both flows by XOR-ing them, which results in 4.66 transmissions in total to deliver one packet for each of the two flows. This yields a gain of 72% over single path routing, 43% over opportunistic routing alone, and 29% over inter-flow coding alone. The *worst case* (*i.e.*, intermediate nodes cannot XOR any packets), which rarely occurs, reverts to opportunistic routing and requires 6.66 transmissions, outperforming single path routing and (worst-case) inter-flow network coding by 20%.

Challenges: The above example demonstrates the potential benefit of inter-flow network coding in opportunistic routing. However, harnessing this gain in prac-

tice poses significant challenges. There exists a *strong tension* between opportunistic routing and inter-flow coding. Opportunistic routing spreads information across multiple nodes. As the information reaching an individual node decreases, the inter-flow coding opportunity decreases because (i) the node itself receives less traffic and has more limited coding choices, and (ii) its next-hops receive less traffic, making it harder to decode. Therefore it is challenging to simultaneously leverage opportunistic forwarding to combat wireless losses and exploit inter-flow coding to reduce traffic.

Our approach: To decouple the strong interactions between opportunistic routing and inter-flow coding, we propose a novel framework to jointly optimize opportunistic routing, rate limiting, and intra- and inter-flow coding. We introduce a novel abstraction by making a wireless network consist of an overlay and underlay, where overlay nodes perform inter-flow coding aware overlay routing without worrying about packet losses and underlay nodes perform intra-flow coding based opportunistic routing without worrying about inter-flow coding.

Our approach: To decouple the strong interactions between opportunistic routing and inter-flow coding, we introduce a novel hierarchical abstraction of overlay and underlay nodes. Here, overlay nodes perform inter-flow coding aware overlay routing without worrying about packet losses and underlay nodes perform intra-flow coding based opportunistic routing without worrying about inter-flow coding.

- *Overlay network:* We designate a subset of nodes as overlay nodes and create an overlay network using them. Each traffic demand is routed over one or more overlay paths. Nodes on the overlay path perform overlay forwarding. They may also use inter-flow network coding to reduce the amount of overlay traffic generated and use inter-flow network decoding to extract the original content.

For example, given two packets, one from flow $f1$ and the other from $f2$, whose overlay paths are $o_1-o_2-o_3$ and $o_3-o_2-o_1$ respectively, node o_2 may XOR the two packets and transmit the inter-coded packet. Nodes o_1 and o_3 perform inter-flow decoding to extract the packets they want. Overlay links are considered reliable so that we can focus on optimizing overlay routes, overlay rate limits, and inter-flow coding without worrying about packet losses.

- *Underlay network:* An overlay link may be mapped to one or more physical links in the underlay network. The underlay network provides efficient and reliable overlay links by using opportunistic routing to spread information across multiple forwarders and letting them cooperatively forward the traffic. To prevent fine-grained coordination, each forwarder independently generates random linear combinations of traffic from the same flow at an appropriate rate so that the destination can extract the original data after receiving enough linearly independent packets. Overlay traffic imposed on each overlay link (whether inter-flow coded or not) is considered as a virtual flow to the underlay network. The goal of an underlay network is to jointly optimize opportunistic routing and rate limiting of the virtual flows without worrying about inter-flow coding.
- *Relationship between the two:* Optimized overlay routing uses efficient overlay routes and inter-flow network coding to reduce the virtual traffic demands imposed on the underlay network. Meanwhile, optimized underlay routing provides efficient and reliable overlay links that the overlay network can take advantage of. The reason that inter-flow coding is put at the overlay network is that the optimization of inter-flow coding is much simpler without packet losses, while opportunistic routing targets packet losses and is naturally to be placed at the underlay network, which involves lossy physical links.

As we see in chapter 3, the key two factors that determine performance of

opportunistic routing in wireless mesh networks are the routes and the rate-limits of the flow sources. Routes determines how effectively Routes determine how effectively we take advantage of communication opportunities and how efficiently we utilize network resources and exploit spatial reuse. Rate limits ensure that traffic sources do not send more than what paths can support. In this chapter, we propose a novel overlay-based optimization framework that effectively combines opportunistic routing, rate limiting, and intra- and inter-flow coding.

In this framework, we formulate the problem of optimizing end-to-end user performance as a linear program (LP) that optimizes total network throughput (or other linear functions) while satisfying: (i) flow conservation constraints for the overlay network, (ii) flow conservation and opportunistic constraints for the underlay network, (iii) constraints that map the traffic demands from the overlay network to the underlay network, and (iv) interference constraints. We then translate the optimization results into practical routing configurations and design an optimized overlay-based opportunistic routing protocol (*O3*) to harness the gains in practice.

We implement *O3* in Qualnet along with (i) shortest path routing (SPP), (ii) SPP with rate limiting, (iii) COPE [67], a state-of-the-art inter-flow coding based routing protocol, (iv) COPE with rate limiting, (v) MORE [27], a state-of-the-art opportunistic routing protocol, and (vi) optimized opportunistic routing, which is also called *O3*-Intra since it is *O3* without inter-flow coding. Using Qualnet simulations, we study the benefits of inter-flow coding, opportunistic routing and rate limiting, and find that (i) rate limiting is important to all routing protocols, (ii) the effectiveness of opportunistic routing increases with loss rates, but the effectiveness of inter-flow coding decreases with loss rates, (iii) *O3* significantly out-performs all the other protocols by simultaneously harnessing the gains of opportunistic routing, inter-flow coding, and rate limiting.

Our main contributions are as follows:

- A novel framework based on the concept of an overlay network to effectively decouple the strong inter-dependency between opportunistic routing and inter-flow network coding.
- The first theoretical formulation that jointly optimizes inter-flow coding, opportunistic routing, and rate limiting. (Section 4.3)
- A practical routing protocol that realizes the optimized opportunistic routes with inter-flow coding and rate limiting. (Section 4.4 and 4.5)
- Extensive evaluation to show the effectiveness of *O3* and the individual benefits of inter-flow coding, opportunistic routing, and rate limiting. (Section 4.6).

4.2 O3 Overview

O3 operates in the following three steps: (i) selecting overlay nodes and overlay paths (Section 4.4.1), (ii) mapping each overlay link into one or more physical links (Section 4.4.1), (iii) jointly optimizing overlay and underlay routing, rate limiting, and inter-flow coding based on the traffic demands, overlay network, and the mapping between the overlay and underlay networks (Section 4.3). The output specifies (i) how fast each source should generate traffic, (ii) how overlay nodes should forward the traffic (*e.g.*, what is the overlay path used, which nodes perform inter-flow coding, and at what rate), and (iii) how underlay nodes should opportunistically forward the traffic (*e.g.*, how many broadcast transmissions to make upon receiving traffic from its neighbor).

In Section 4.3, we first present an optimization framework for (iii), which takes overlay paths and mappings between overlay and underlay networks as input

and outputs the optimized overlay routing, underlay routing, rate limits, and inter-flow network coding. The output is optimal when the input enumerates all possible overlay paths and maps each overlay link to the entire underlay network (*i.e.*, lets each overlay link use any underlay link for potential routing). However, this optimization problem may incur significant computation cost due to a large number of optimization variables. In Section 4.4, we describe our approach to improve scalability. It reduces the size of the optimization problem by selectively choosing overlay paths and mapping each overlay link to a small subset of underlay links.

Before delving into the details of each step, let us first go through a simple example shown in Figure 1.1, which has two flows in opposite directions. Suppose we select nodes A and D as overlay nodes; meanwhile we choose AD as an overlay path for flow 1 and choose DA as an overlay path for flow 2. Then in the overlay network, node A sends to node D via the overlay path AD, and node D sends to node A via the overlay path DA. There is no inter-flow coding since there is no intermediate overlay node in this case. If the overlay link AD is mapped to the entire physical network as an underlay, the underlay network is responsible for sending traffic for flow f1 from node A to node D using opportunistic routing on the entire underlay network. Similarly, if the overlay link DA uses the entire physical network as the underlay, then the corresponding underlay network is responsible for sending flow f2 from node D to node A using opportunistic routing. So essentially each underlay network tries to carry the traffic imposed by the corresponding overlay link l from $src(l)$ to $dest(l)$, where $src()$ and $dest()$ denote the source and destination of the link, respectively.

Alternatively, we may select nodes A, B, C, D as overlay nodes, and choose AD, ABD, ACD as overlay paths for flow 1 and DA, DBA, DCA as overlay paths for flow 2. Then in the overlay network, node A splits its traffic across the three overlay

paths according to the optimization output. So does node D. Node B may XOR flow f1's traffic sent on ABD with flow 2's traffic sent on DBA, and the fraction of inter-flow coded traffic is determined by the optimization output. Similarly for node C. As before, the underlay network is responsible for opportunistically routing all the traffic imposed by the corresponding overlay link, where the imposed traffic can be either inter-flow coded or not.

4.3 O3 Problem Formulation

Based on the overlay framework, we derive a linear program (LP) that consists of the following four components: (i) flow conservation constraints on the overlay network that can use inter-flow coding, (ii) flow conservation and opportunistic constraints on the underlay network that uses intra-flow coding based opportunistic routing, (iii) constraints mapping traffic demands from the overlay network to the underlay network, and (iv) interference constraints to prevent interfering links from being active simultaneously. The key challenge in this formulation is to accurately capture virtual flows and physical flows and interactions between the overlay and underlay networks. Below we describe the formulation in detail.

4.3.1 Optimization Objective

Our framework is general and can optimize any linear function. We focus on the most common metric: maximizing total throughput, namely $\sum_{k \in D} \sum_{P \in PS^k} f^k(P)$, where D is the set of traffic demands, $f^k(P)$ is the k-th flow's throughput over path P, and PS^k is the set of paths used by the k-th flow. Alternatively, we can support (i) maximizing a linear approximation of proportional fairness, defined as $\sum_{k \in D} log(\sum_{P \in PS^k} f^k(P))$, which strikes a good balance between fairness and throughput [119], (ii) maximizing the fraction of demand that is served from each

flow, denoted as α, where $\alpha \cdot D_k$ is the lower bound of throughput for the k-th flow, or (iii) maximizing total revenue if the revenue is a linear function of throughput.

4.3.2 Overlay Network Constraints

The route on an overlay network must satisfy flow conservation. We derive the flow conservation constraints by applying coding-aware optimization for single path routing, as described in [127]. The main difference from traditional flow conservation is inter-flow coding allows an intermediate node to deliver different information to different neighbors using the same transmission. Therefore we need to classify traffic into native (*i.e.*, without inter-flow coding) and inter-coded, and derive the constraints based on the traffic type. Specifically, let $z_i^k(P)$ denote the amount of native traffic transmitted by node i for flow k over path P. Let $x_i(e1, e2, n)$ denote the amount of traffic received from link $e1$ as native traffic and transmitted by node i over link $e2$ as inter-flow coded, and $x_i(e1, e2, c)$ denote the amount of traffic received from link $e1$ as inter-flow coded traffic and transmitted by node i over link $e2$ as inter-flow coded traffic. We call $(e1, e2, n)$ and $(e1, e2, c)$ *coding structures*. Let CS denote the set of coding structures in the network. We have the following flow conservation constraints under inter-flow network coding, where $e1$ is node i's incoming link and $e2$ is node i's outgoing link.

- $\sum_{(e1,e2,n)\in CS} x_i(CS) \leq \sum_{k\in D} \sum_{e1e2\in P, P\in PS^k} z^k_{t(e1)}(P)$. This says that the transit traffic participating in coding as *native-received* at node i is bounded by the total native traffic received from $t(e1)$, which is the transmitter of link $e1$.
- $\sum_{(e1,e2,c)\in CS} x_i(CS) \leq \sum_{k\in D} \sum_{e1e2\in P, P\in PS^k} [f^k(P) - z^k_{t(e1)}(P)]$. This reflects that the total traffic that participates in coding as *coded-received* at node i is bounded by the amount of traffic received as coded at node i.

- $\sum_{k \in D} \sum_{e1e2 \in P, P \in PS^k} f^k(P) = \sum_{k \in D} \sum_{e1e2 \in P, P \in PS^k} z_i^k(P) + \sum_{(e1,e2,n) \in CS} x_i(CS) + \sum_{(e1,e2,c) \in CS} x_i(CS)$. This indicates the total traffic received from link $e1$ and transmitted over link $e2$ by node i must be one of the three types of traffic: (i) traffic going out as native, (ii) traffic participating in coding as native-received, and (iii) traffic participating in coding as coded-received.
- $z_{src(k)}^k(P) = f^k(P)$, where $P \in PS^k$. This indicates a flow source $src(k)$ transmits all traffic as native over every path.
- $z_i^k(P) \leq f^k(P)$, where $i \in P - \{src(k), dst(k)\}$ and $P \in PS^k$. This indicates that the amount of native traffic transmitted by a transit node is bounded by the total traffic on the path P.

4.3.3 Underlay Network Constraints

The goal of the underlay network is to use opportunistic routing to efficiently and reliably route the traffic demands imposed by the overlay network. The traffic includes either an original flow f or inter-flow coded traffic between multiple flows. For convenience, we denote either original or inter-flow coded traffic as *physical flow* pf. Then every combination of overlay link vl and physical flow pf is considered as a *virtual flow*, denoted by (vl, pf). For example, consider 3 physical flows in the overlay network: $f1, f2, f1 + f2$. The virtual traffic demands on the underlay network are $< o_i - o_j, f1 >, < o_i - o_j, f2 >, < o_i - o_j, f1 + f2 >$, where $o_i - o_j$ denotes any overlay link. Let $src(vl)$ and $dest(vl)$ denote the source and destination of the overlay link vl. The underlay network uses optimized opportunistic routing to efficiently route the physical flow pf from $src(vl)$ to $dest(vl)$.

Underlay flow conservation constraints: To ensure valid opportunistic routes on the underlay, we first derive flow conservation constraints for each virtual flow. Dif-

ferent from traditional flow conservation, the flow conservation constraints of the underlay only apply to the amount of information (*i.e.*, non-redundant useful data) instead of traffic due to packet losses. Let $Y(vl, pf, i, j)$ denote the information transmitted from node i to node j for the virtual flow (vl, pf).

- $Y(vl, pf, k, src(vl)) = 0$ for any node k. This enforces no incoming information to $src(vl)$ for a virtual flow (vl, pf) since $src(vl)$ is the source of the virtual flow.
- $Y(vl, pf, dest(vl), k) = 0$ for any node k. This enforces no outgoing information from $dest(vl)$ for a virtual flow (vl, pf) since $dest(vl)$ is the destination of the virtual flow.
- For any transit node $i \neq src(vl)$ and $i \neq dest(vl)$, $\sum_{k \in in(i)} Y(vl, pf, k, i) \geq \sum_{j \in out(i)} Y(vl, pf, i, j)$, where $in(i)$ and $out(i)$ denote node i's incoming and outgoing neighbors, respectively. It ensures that the incoming information to node i is no less than the outgoing information from i.
- $\sum_k Y(vl, pf, src(vl), k) \leq NR(vl, pf)$. This denotes that the amount of information successfully delivered from $src(vl)$ to node k is bounded by the virtual flow's traffic demand, denoted as $NR(vl, pf)$.

Underlay opportunistic constraints: Next we capture the relationships between the amount of traffic and the amount of information delivered on the underlay network. We formulate these relationships using the following opportunistic constraints, where the first one captures the relationships for a given virtual flow while the next two constraints capture the relationships for a physical flow that spans multiple overlay links from the same overlay source. The latter constraints are necessary because we allow an overlay source to broadcast traffic over multiple overlay links simultaneously and let all its downstream nodes derive information from the

same transmission. Therefore we need to ensure the total information derived across all overlay links and across all downstream nodes does not exceed the amount of successfully received traffic.

- Virtual flow opportunistic constraint: $S(i, \mathcal{N}(i))T(vl, pf, i)$ $\geq \sum_{k \in \mathcal{N}(i)} Y(vl, pf, i, k)$, where $\mathcal{N}(i)$ denotes a subset of i's neighbors, $S(i, \mathcal{N}(i))$ is the probability of successfully delivering traffic to any node in $\mathcal{N}(i)$, and $T(vl, pf, i)$ is the amount of traffic transmitted from node i on overlay link vl for flow pf. This constraint indicates for any virtual flow (vl, pf) the total traffic successfully delivered to at least one neighbor in $\mathcal{N}(i)$ should be no less than the total amount of non-overlapping information delivered to $\mathcal{N}(i)$. When i has many (say, K) neighbors, enumerating $\mathcal{N}(i)$, all subsets of neighbors, is costly. For scalability, when $K > 3$, we enumerate the neighbor sets of size 1, size 2, and the one containing all i's neighbors (*i.e.*, enumerate only $O(K^2)$ instead of $O(2^K)$ neighbor sets).
- Physical flow opportunistic constraint 1: $S(i, k)MaxT(pf, i)$ $\geq \sum_{(i,*) \in vl} Y(vl, pf, i, k)$, where $MaxT(pf, i)$ is the total overlay traffic node i sends for physical flow pf over all overlay links. Due to the broadcast nature of overlay traffic (*i.e.*, an overlay node can use a single transmission to send a packet along multiple overlay paths by including all the overlay paths in the packet header), $MaxT(pf, i) = max_{vl} T(vl, pf, i)$. These constraints together enforce that total information delivered from i to k over all virtual links is bounded by the total traffic successfully delivered from node i to k for the physical flow pf.
- Physical flow opportunistic constraint 2: This constraint further ensures that the total amount of information delivered to a subset of i's neighbors, denoted as $\mathcal{N}(i)$, over all virtual links is bounded by the product of i's traffic and the

probability of successfully delivering to at least one neighbor in $\mathcal{N}(i)$:

$$S(i, \mathcal{N}(i)) MaxT(pf, i) \geq \sum_{k \in \mathcal{N}(i)} \sum_{(i,*) \in vl} Y(vl, pf, i, k).$$

To improve scalability, we use the same enumeration procedure as in constructing the virtual flow opportunistic constraints (*i.e.*, enumerating the neighbor sets of size 1, size 2, and the one containing all i's neighbors when $K > 3$).

4.3.4 Constraints Relating Overlay to Underlay

To relate the overlay to the underlay network, we derive the following constraints. The first two constraints relate the traffic demands of the virtual flow with the overlay traffic, and the last constraint ensures the virtual flow is serviced by the underlay network:

- $NR(vl, pf) = \sum_{vl \in P} z^k_{src(vl)}(P)$, where $pf = (native, k)$. This reflects that the traffic demand for a native flow (vl, pf), denoted as NR, is equal to the amount of native traffic flow k sent over the virtual link vl.
- $NR(vl, pf) = \sum_{vl \in P} x_{src(vl)}(CS)$, where CS is the coding structure and $pf = (coded, CS)$. This indicates that the traffic demand for a coded virtual flow (vl, pf) is equal to the coded traffic sent using the same coding structure.
- $NR(vl, pf) = \sum_{k \in in(dest(vl))} Y(vl, pf, k, dest(vl))$, which indicates that the traffic demand for the virtual flow (vl, pf) is honored by the underlay network, *i.e.*, the traffic demand $NR(vl, pf)$ is successfully delivered to $dest(vl)$.

4.3.5 Interference Constraints

Finally, we impose interference constraints for the traffic sent on the physical network, since this is the actual traffic transmitted. Based on the network

topology, we construct a broadcast conflict graph. Specifically, two transmitters are considered to have conflict if either of the following conditions holds: (i) the two transmitters are within carrier sense range of each other, or (ii) one receiver is within the interference range of the other transmitter. We then find independent sets in the conflict graph and derive the following interference constraints that indicate the total activity time of a node is no more than the sum of activity time of all the independent sets that the node belongs to.

- Let MT_i denote the total traffic from node i. If node i is an overlay node, we have $MT_i = \sum_{pf} max_{vl} T(vl, pf, i)$; otherwise we have $MT_i = \sum_{pf} \sum_{vl} T(vl, pf, i)$. The reason for such a distinction is that the overlay node uses the broadcast nature of the wireless medium to transmit over multiple overlay links simultaneously by including these overlay links in its packet header. In comparison, underlay nodes forward for a specific overlay link and thus an underlay node needs to separately forward for each overlay link included in the received packet's header.
- For every node i, $MT_i \leq Cap_i \sum_{k \in I_i} \lambda_k$, where Cap_i is node i's broadcast data rate, I_i denotes the independent sets that node i belongs to, and λ_k denotes the activity time of independent set k. This constraint enforces the total traffic sent by any node is bounded by the sum of the activity time of the independent sets that the node belongs to scaled by the wireless capacity.
- $\sum_k \lambda_k \leq 1$ because only one independent set can be active at a time.

4.4 Leveraging the Optimization Framework

In this section, we describe how to obtain the inputs required by the optimization and how to translate the optimization results into routing configurations.

4.4.1 Obtaining Inputs

Our optimization algorithm requires the following inputs: network topology, traffic demands, overlay paths, and mapping from overlay to underlay network resources. The network topology can be obtained easily through periodic measurements. As reported in [44, 79], wireless traffic exhibits temporal stability, and we can estimate current traffic demands based on previous demands. Thus, here we focus on the latter two inputs. Our optimization framework in Section 4.3 is flexible and can easily take inputs generated by other overlay path selection and overlay-to-underlay mapping algorithms.

Selecting overlay nodes: One way to select overlay nodes is to let every physical node serve as an overlay node. This leads to the best performance at the cost of higher computation time, since the computation cost increases with the number of overlay nodes. Therefore we want to limit the number of overlay nodes. Since only intermediate overlay nodes perform inter-flow coding, our goal is to select overlay nodes with high coding opportunities.

To achieve this goal, for each flow f_k we order the nodes on its forwarding list according to the coding opportunities. We estimate the upper-bound of the coding opportunities as given by

$$min(T(f_k, i), \sum_{j \in D}(min(T(f_k, i), T(f_j, i)))), \tag{4.1}$$

where $f_k \neq f_j$ and $T(f_k, i)$ is total traffic transmitted by node i for flow f_k. Equation (4.1) is derived based on the fact that the rate of inter-flow traffic between two flows is bounded by the minimum rate of these two flows. Therefore, $min(T(f_k, i), T(f_j, i))$ gives an upper-bound on the amount of traffic that can be inter-flow coded between f_k and f_j, and Equation (4.1) gives an upper-bound of total traffic that can be inter-flow coded between f_k and all the other flows. For every

flow, we pick the top three nodes from the sorted list as the overlay nodes. When the upper-bound is the same, we use the amount of traffic sent in either direction to break ties.

Selecting overlay paths: After selecting overlay nodes, we then generate overlay paths for each flow. Each flow contains at least one overlay path directly from the source to the destination, and this overlay path is mapped to the entire underlay network to ensure the solution is no worse than opportunistic routing alone, which is a special case of *O3*. If this is the only overlay path between the source and destination, *O3* becomes opportunistic routing alone, since this overlay path does not involve an intermediate node and there is no inter-flow coding.

To leverage inter-flow coding, a flow may contain other overlay paths going through one or more intermediate nodes. For each flow, we identify the overlay nodes (selected in the previous step) that are on the flow's forwarding list, which includes the flow source and destination. We enumerate all possible overlay paths involving these nodes, where their order on the overlay path is based on their ETX [31] (*i.e.*, the number of required transmissions to deliver a packet) to the destination.

Mapping overlay network to underlay network resources: The goal of this step is to map each overlay link to one or more physical links. Only the physical links, to which the overlay link is mapped to, can potentially be used as part of an opportunistic route; but whether these physical links actually participate in opportunistic routing and how much traffic they each route depend on the optimization result of the problem formulated in Section 4.3.

One possible mapping is to let each overlay link span all physical nodes and links. To enhance scalability, we treat an overlay link $o1 - o2$ as a virtual traffic

demand and use MORE to select nodes and links to be included in the underlay network. Specifically, we find the forwarding list for this virtual flow from $o1$ to $o2$ using MORE. The overlay link then uses all nodes on the forwarding list as underlay nodes, and uses physical links between these nodes as underlay links. The intuition behind this mapping is that links on the opportunistic routes are most useful for forwarding traffic from $o1$ to $o2$.

4.4.2 Executing Optimization

The optimization can run at a central location that distributes the optimization results to all nodes. The amount of information to distribute is small compared to data traffic. Specifically, the input includes traffic demands, link loss rates, and the conflict graph, which are $O(F)$, $O(E)$, $O(E^2)$, respectively, where F is the number of flows and E is the number of physical links. Among these three terms, $O(E^2)$ is a dominating term, so the input requires $O(E^2)$. The output includes overlay and underlay credits, which are $O(ON \cdot F \cdot P)$ and $O(N \cdot D \cdot F \cdot OE) + O(N \cdot D \cdot OE^2)$, respectively, where ON is the number of overlay nodes, N is the number of physical nodes, P is the number of overlay paths, D is the number of physical neighbors, and OE is the number of overlay links. Therefore we can tradeoff between the wireless performance and the size of information to be exchanged by controlling the number of overlay nodes and links. Moreover, only non-zero credits need to be exchanged. From our experience, a large majority of credits are zero so the actual information to be exchanged is well below the above worst case (*e.g.*, only a few KB for a 25-node network in our simulation).

Instead of centralized computation, the computation can be done in a distributed fashion, similar to link-state protocols like OSPF [100], where every node implements the same algorithm over the same data to arrive at the same results. The

amount of link state information is very small. The optimization is executed either periodically or upon changes in network topology or traffic conditions. The computation time is reasonable (*e.g.*, around 3.6 seconds for 4 flows in 25-node random networks used in our evaluation). To further enhance scalability, when the inputs change slightly, we can leverage incremental LP solvers, such as lp_solve_inc [55], to take advantage of incremental changes in the linear constraints and more efficiently derive a solution to the new LP rather than solving it from scratch.

In addition to optimization based on the global information, as part of our future work, we are interested in applying decomposition techniques developed for distributed convex optimization (*e.g.*, [68]) to solve the optimization based on decentralized information to further enhance the scalability.

4.4.3 From LP Output to Routing Configurations

The optimization results specify the desired sending rates for both inter- and intra-flow coded traffic. A flow source i transmits at the rate of $max_{vl}T(vl, pf, i)$ for its flow pf. An intermediate node uses a credit-based scheme to enforce its forwarding strategy according to the derived $T(vl, pf, i)$, where pf can be either inter-flow or intra-flow coded. Specifically, underlay nodes do not care about inter-flow coding and simply forward traffic pf according to $T(vl, pf, i)$. Overlay nodes perform inter- and intra- encoding and decoding as specified in Section 4.5.1. Since the exact rate of sending inter-flow coded traffic at an overlay node depends on traffic dynamics and is hard to enforce, we convert the desired traffic rates into intra-flow credits and use inter-flow coding whenever an opportunity arises. Note that our credit computation is different from [27] due to significant difference in the two routing protocols (*e.g.*, *O3* needs to compute overlay and underlay credits, whereas [27] has only one type of credit). Below we specify credit computation for

underlay and overlay nodes based on the LP output.

The credit is defined as the number of transmissions that should be generated for every received packet. Upon receiving a packet, a node increments its credit. When this credit becomes greater than or equal to 1, it generates a transmission and then decrements its credit by 1. This process is repeated until its credit goes below 1. Based on this credit definition, we can compute the credit as the total desired sending rate divided by the total receiving rate. Credit information is then stored as the following tuples: $(f, P, i, credit)$ for overlay nodes, $(f, vl, prev(i), i, credit)$ for underlay nodes' intra-coding credits, and $(vl1, vl2, prev(i), i, credit)$ for underlay nodes' inter-coding credits, where f is the flow id, P is the overlay path id, i is the node id, vl is the overlay link id, $prev(i)$ is the previous hop of node i in the underlay network, $vl1 - vl2$ is the overlay segment and $prev(i) - i$ is an underlay link that is responsible for forwarding traffic for the overlay segment.

We first compute underlay credits. Upon receiving a transmission from node j, underlay node i increments its credit by $C \times R$, where C reflects the fraction of useful information contained in each transmission from node j, and R reflects the amount of redundancy node i should include to compensate for loss to its forwarders. Therefore, we have $C = Y(vl, pf, j, i)/(TC(vl, pf, j) * (1 - loss(j, i))$, where its numerator is the amount of information received and its denominator is the amount of traffic received, and their ratio gives the amount of information contained in a received packet. $R = T(vl, pf, i)/\sum_k Y(vl, pf, i, k)$. R's numerator is the desirable sending rate, its denominator is the total information successfully delivered to its forwarders k's, and their ratio indicates how much traffic to generate in order to deliver one-packet worth information to i's forwarders.

Next we compute overlay credits. Upon receiving intra-flow coded traffic,

an overlay node i increments its credit for a given path P and flow f by

$$T(vl, f, i) * \frac{z_i^{pf}(P)}{\sum_{P_i: vl \in P_i} z_i^{pf}(P_i)},$$

where the second term in the product is how much fraction of native traffic node i received along virtual link vl is for path P, and the product indicates the total amount of native traffic received over vl for path P. Upon receiving inter-flow coded traffic, an overlay node increments its credit associated with the intra-flow involved by $(z_{src(pf)}^{pf}(P) - z_i^{pf}(P)) * NSR(i, vl)$, where the first term in the product indicates how much inter-flow coded information is at node i and $NSR(i, vl)$ is the expected number of transmissions required to successfully deliver a packet to one of i's forwarders and can be computed as

$$1.0/(1.0 - \prod_{\forall k \in fwd(i)} loss(i, k)),$$

assuming independent packet losses at different nodes, where $fwd(i)$ denotes node i's forwarding list. For example, if node i is an overlay forwarder for $f1$, upon receiving $f1 + f2$, it increments $f1$'s credit as described above.

4.5 Protocol Specification

We now describe how to achieve a practical routing protocol, *O3*, based on the optimization results. In this section, we first present the algorithm to perform joint inter-flow and intra-flow encoding and decoding, and then describe the behaviors of flow sources, destinations, and forwarders.

4.5.1 Packet Coding Algorithm

We use random linear coding to code packets within the same flow and use XOR to code packets across flows. In our implementation, we inter-code up to 2

flows, the common case for inter-coding. Our coding algorithm is general and can code more flows at a higher computational cost. Below we present the detailed algorithms.

Encoding: To code intra-flow data, a flow source $src(f)$ divides user traffic into batches, as in MORE. Each batch has K packets, where K is a tunable parameter to trade-off between batching overhead and delay. When the MAC is ready for transmission, $src(f)$ or its forwarder, generates a random linear combination of all packets it has from the current batch and broadcasts this packet. We refer to such a coded packet as an *intra-coded* packet. To code inter-flow packets from two batches, denoted as $(f1, b1)$ and $(f2, b2)$, a node first generates an intra-coded packet P_1 using a random linear combination of all packets in $(f1, b1)$, and similarly generates packet P_2 from $(f2, b2)$. Then it XORs packets P_1 and P_2 to create an *inter-coded* packet.

Decoding: Each incoming packet yields a linear constraint. If the incoming packet is intra-coded from batch $(f1, b1)$ with batch size of $K1$, the constraint involves $K1$ variables in $(f1, b1)$. If the incoming packet is inter-coding of $(f1, b1)$ and $(f2, b2)$, whose batch sizes are $K1$ and $K2$, respectively, this inter-coded packet gives one constraint involving $K1 + K2$ variables for these two batches.

The goal of intra-flow decoding is to recover the original packets from the batch. For the batch size of $K1$, a node can use Gaussian Elimination to decode the entire batch when it has $K1$ innovative (*i.e.*, linearly independent) packets.

The goal of inter-flow decoding is to extract intra-coded packets, which in turn can be used to extract the original packets from the batch. For example, if a node has everything from $(f1, b1)$, then reception of an innovative inter-coded packet with $(f1, b1) + (f2, b2)$ (*i.e.*, linearly independent of the other inter-coded

packets) allows us to extract one intra-coded packet for $(f2, b2)$ using Gaussian Elimination. More generally, if the inter-coding matrix has rank r, then we can use Gaussian Elimination to extract $max(r - K1, 0)$ intra-coded packets for batch $(f2, b2)$, and extract $max(r - K2, 0)$ intra-coded packets for batch $(f1, b1)$.

To support intra- and inter- decoding, a node maintains intra- and inter-coding matrices, which store the coefficients used in all the innovative packets. The main design issue in the decoding algorithm is how to handle interactions between the intra-coding and inter-coding matrices. To simplify the encoding and decoding processes, we maintain all the information in the intra-coding matrix if there is

```
1  upon receipt of a packet pkt:
2  if(pkt is intra-coded)
3    (f,b) = extract-flow-and-batch(pkt);
4    all-matrices = find-all-matrices-containing(f,b);
5    if (all-matrices == NULL)
6       new-matrix = create-matrix(f,b);
7       add pkt to new-matrix;
8    else
9         foreach matrix ( all-matrices )
10           add pkt to matrix
11        end
12   end
13 else // inter-coded
14      (f1,b1,f2,b2) = extract-flow-and-batch(pkt);
15      inter-matrix = find-matrix(f1,b1,f2,b2);
16      intra-matrix1 = find-matrix(f1,b1);
17      intra-matrix2 = find-matrix(f2,b2);
18      if (inter-matrix == NULL)
19         new-matrix = create-matrix(f1,b1,f2,b2);
20         add intra-matrix1 to new-matrix;
21         add intra-matrix2 to new-matrix;
22         add pkt to new-matrix;
23      else
24          add pkt to inter-matrix;
25          v = extract-pkt-from-matrix(inter-matrix,f1,b1);
26          if (v != NULL)
27             add v to intra-matrix1;
28          end
29          v = extract-pkt-from-matrix(inter-matrix,f2,b2)
30          if (v != NULL)
31             add v to intra-matrix2
32          end
33      end
34 end
```

Figure 4.1: Decoding algorithm

no inter-coding matrix involving the batch; otherwise we keep information in both intra-coding and inter-coding matrices. We extract intra-coding constraints from the inter-coding matrices whenever possible and add it to the corresponding intra-coding matrix. Specifically, when a node receives a packet, it uses the packet header to determine whether it is intra-coded or inter-coded. An intra-coded packet should be added to the intra-coding matrix involving the batch to which the packet belongs, as well as to the inter-coding matrix, if the batch is involved in inter-coding. An inter-coded packet is first added to the inter-coding matrix, from which we extract an intra-coding constraint if its rank is large enough (*i.e.*, exceeding either $K1$ or $K2$). If the packet is the first inter-coded packet for the batch pair, we (i) create an inter-coding matrix, (ii) copy the intra-coding matrices to the inter-coding matrix if one or more exist (so that the inter-coding matrix maintains all the intra-flow information obtained so far), and (iii) add the new packet to the inter-coding matrix. Figure 4.1 shows the pseudo code of our decoding algorithm.

To reduce storage cost, we identify active batches as described in Section 4.5.2 and store coding matrices only for the active batches. The intra-coding matrix can be removed immediately when the corresponding batch becomes inactive, while the inter-coding matrix can be removed only when both batches in the matrix become inactive. In our evaluation, storage per node is 300 KB for 16 flows in a 25-node random topology spanning 1000x1000 m^2, which is easily affordable for today's hardware.

4.5.2 Flow Sources and Destinations

A flow source, $src(f)$, $never$ performs inter-flow encoding and only generates intra-coded packets at the rate computed by the LP. Each packet generated by the $src(f)$ or intermediate forwarders includes all the overlay paths that the packet

may traverse and the current overlay link associated with each overlay path. Further, to facilitate the decoding of any inter-coded packets in the future, $src(f)$ saves the intra-coded packet it transmits in its buffer until the corresponding batch becomes inactive.

In MORE, $src(f)$ continues transmitting packets from the current batch until it receives an ACK for the batch. This incurs significant stop-and-wait overhead. To reduce such overhead, a large batch size K would be beneficial. However, to effectively support inter-flow coding, we prefer a small batch size, since a node can start extracting a new intra-coded packet only when the rank of the inter-coding matrix exceeds K. The larger the value of K, the lower the inter-flow coding opportunity. To efficiently support a smaller batch size, we allow a flow source to send multiple batches before receiving an ACK. The destination generates an ACK either when an entire batch is received or when a threshold number of new packets are received since the last ACK. The ACK contains *(min-active-batch-id, active, status)*, where *min-active-batch-id* is the id of the smallest active batch, $active$ is a bit map where $active[i] = 1$ indicates batch i is active and has not been ACKed, and status is an array indicating the number of innovative packets received by the destination for each active batch. The source uses this information to schedule transmissions from different batches in a FIFO order, and the forwarders use the information to remove inactive batches.

4.5.3 Forwarders

In this subsection, we describe two major tasks of a forwarder: (i) processing a received packet and (ii) generating and transmitting a packet when the medium is available.

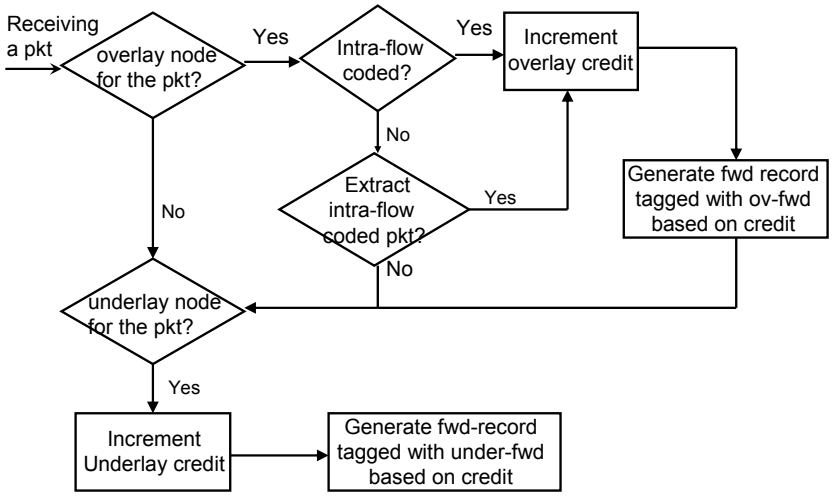


Figure 4.2: Steps involved in processing a received packet at an intermediate node.

4.5.3.1 Process a received packet

Determine whether to perform overlay and/or underlay forwarding:

As shown in Figure 4.2, upon receiving a packet, a node first checks if it is an overlay and underlay forwarder for this packet. It does so by inspecting the set of overlay paths/links included in the packet. A node is an overlay forwarder for an *intra-flow* coded packet if it is on at least one of the overlay paths in the packet header, and is an overlay forwarder for an *inter-flow* coded packet if it is an overlay forwarder for either $f1$ or $f2$. In either case, it invokes overlay forwarding operation. A node then checks if it is an underlay forwarder for this packet in a similar way. If it is, it invokes underlay forwarding operation. Note that it is possible for a node to perform both overlay forwarding and underlay forwarding for the same packet. If a node is neither an overlay nor underlay forwarder for a packet, it simply drops the packet.

Overlay node operation: An overlay forwarder is responsible for forwarding traf-

fic to the next overlay node along the overlay path and performing inter-flow encoding and decoding whenever necessary. For an overlay node, if it receives an intra-flow coded packet, it looks up its credit table computed based on the optimization results as described in Section 4.4.3 to determine how many packets to send. Instead of generating actual packets for transmission, it generates forwarding records (one for each packet to be sent out), where the record specifies the flow, overlay path(s), batch, and the forwarding mode of the packet (*e.g.*, whether *ov-fwd* or *under-fwd*). The actual packets are not generated until the medium becomes available for transmission. Delaying packet generation until transmission allows us to make up-to-date intra-coding and inter-coding decisions.

It then tags the generated records with *ov-fwd* to indicate they are eligible for inter-flow coding, and inserts them into the queue, which will result in packet generation and transmission when the medium becomes available.

If it receives an inter-flow coded packet $P(f1, b1, f2, b2)$, it first checks whether it can extract an intra-coded packet of $(f1, b1)$ or $(f2, b2)$. If so, this reduces to the case of receiving an intra-coded packet. Otherwise, it inserts the coding coefficient into the corresponding inter-coding matrix and waits for future extraction of an intra-coded packet. This wait time is bounded by a threshold, after which the packet is garbage collected.

Underlay node operation: The goal of an underlay forwarder is to forward traffic for the current overlay link using opportunistic routing. It looks up its corresponding credit increment table as computed in Section 4.4.3, generates forwarding records according to the credit, tags each record with *under-fwd* (to prevent them from performing inter-flow coding), and inserts them into the output queue. Note that the processing is similar for inter-flow and intra-flow coded packets. The only

difference is that a different credit table is consulted to determine the number of forwarding records to generate.

Generating forwarding records: To handle multiple outstanding batches and multiple overlay paths per flow, a forwarder not only maintains a flow credit $\mathit{flow}_{cr}(f, p)$ for each combination of flow f and overlay path p but also maintains a batch credit $batch_{cr}(b, f, p)$, where b is the batch id. flow_{cr} determines the transmission rate for a given flow over a given overlay path, and $batch_{cr}$ determines the transmission rate for a specific batch. When receiving a packet, we update all the flow and batch credits that match f, b and $\forall p \in$ *ov-set*. A node generates a record for flow f as long as $max_p(\mathit{flow}_{cr}(f, p)) \geq 1$. The record includes all overlay paths p_i with $\mathit{flow}_{cr}(f, p_i) \geq 1$. To handle multiple batches, a record is generated from the batch with the largest batch credit over all overlay paths, *i.e.*, the largest $\sum_{p \in OS} batch_{cr}(b, f, p)$, where OS is the set of overlay paths that the current packet should be sent along. After constructing the record, an overlay forwarder tags it with *ov-fwd* whereas an underlay forwarder tags it with *under-fwd*. In both cases, the forwarder inserts the generated record to the FIFO queue, decrements the corresponding flow_{cr} and $batch_{cr}$ values by 1, and continues generating new forwarding records until the flow_{cr} drops below 1.

4.5.3.2 Transmit when medium becomes available

When the medium is available, the node dequeues forwarding records from its queue and generates a corresponding packet for transmission. More specifically, the forwarder dequeues the first forwarding record *(f1,b1,ov-set1)* from its queue and if the record is tagged with *under-fwd*, it generates a random linear combination of all packets corresponding to flow $f1$ and batch $b1$ and transmits it. If the record is tagged with *ov-fwd*, which indicates it is eligible for inter-flow coding, it searches

for another record from its queue *(f2,b2,ov-set2)* that can be inter-flow coded with the first packet. If a match is found, the node dequeues *P(f2,b2,ov-set2)*, inter-flow codes the two packets, and includes (*ov-set1*, *ov-set2*) in the packet header to indicate the packet should be forwarded along the paths in *ov-set1* and *ov-set2*. Then it broadcasts the resulting inter-flow coded packet. If no match is found, the nodes generate an intra-coded packet from flow $f1$ and batch $b1$, includes *ov-set1* as the overlay path, and sends it out.

To check if two packets can be inter-flow coded, we examine the positive x_i (defined in Section 4.3.2) values from the LP to determine the combinations of overlay nodes and overlay paths that are involved in inter-flow coding. We store these positive values in a lookup table at each node. Two packets $P1$ and $P2$, containing the set of overlay paths *ov-set1* and *ov-set2*, respectively, can be inter-flow coded if and only if for each $ov1 \in$ *ov-set1* and $ov2 \in$ *ov-set2*, there exists an entry in the lookup table indicating we can inter-code $ov1$ and $ov2$.

To enhance inter-flow coding opportunity, we introduce two queues Q_{inter} and Q_{intra}, where packets from Q_{intra} are usually sent out as intra-flow coded, and packets from Q_{inter} are sent out as inter-flow coded *whenever* possible. Based on the LP output, we compute the ratio of inter-flow versus intra-flow coded traffic, and insert packets into these queues according to these ratios. We also associate a timeout with every packet in Q_{inter}. Once the medium is available for transmission, we poll the first packet from Q_{inter}, denoted as P, and searches for another packet to inter-code with P first from Q_{inter} and then from Q_{intra}. If found, we send out the resulting inter-flow coded packet immediately. Otherwise if P's associated timer has not expired, we instead send out the first packet from Q_{intra}. When the timer expires, we send out P even if it cannot be inter-flow coded with another packet to limit its delay.

4.6 Performance Evaluation

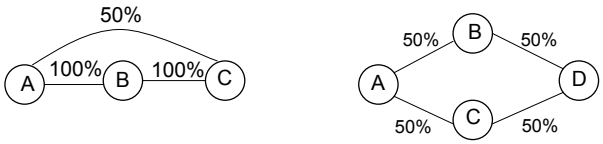
In this section, we first describe our evaluation methodology, and then present performance results.

4.6.1 Evaluation Methodology

We implement *O3* and the following protocols in Qualnet 3.9.5 and conduct extensive simulation to compare their performance:

1. Shortest-path routing (SPP) using the ETX routing metric, which minimizes the total number of expected transmissions from a source to its destination [31].
2. Shortest-path routing with rate-limiting (SPP-RL), the same as SPP except the flows' sending rates are optimized using the conflict graph interference model as in [79].
3. COPE, a state-of-art shortest path routing protocol with inter-flow network coding.
4. COPE with rate limiting (COPE-RL), the same as COPE except that the flows' sending rates are optimized using the conflict graph model.
5. MORE, a state-of-art opportunistic routing protocol.
6. Optimized opportunistic routing, also called *O3*-Intra, since it is the same as *O3* except that it disables inter-flow coding.

O3-Intra improves MORE by optimizing opportunistic routing and rate limiting. To our knowledge, this is the first work that extensively compares single path routing, opportunistic routing, and inter-flow coding with and without rate limiting. The evaluation allows us to not only understand the performance of *O3* but also examine individual benefit of inter-flow coding, opportunistic routing, and rate limiting.



(a) Linear chain (b) Diamond

Figure 4.3: Two symmetric flows between the left-most and right-most nodes.

Since SPP uses unicast transmissions, SPP-RL uses a link-based conflict graph model, which represents wireless links as vertices in a conflict graph and draws an edge between two conflict vertices if and only if the corresponding wireless links interfere. Based on this definition, links corresponding to the vertices in a clique of the conflict graph cannot be active simultaneously. Since COPE and all of the opportunistic routing protocols use either pseudo or real broadcast transmissions, we use a node-based conflict graph model, which considers two broadcast transmissions to interfere if either (i) the transmitters carrier sense each other or (ii) anyone of their receivers is interfered by the other transmission.

Both MORE and *O3*-Intra use a batch size of 32 packets, which is the default batch size used in MORE [27]. Further increasing the batch size yields little benefit. *O3* uses a batch size of 16 with 2 outstanding batches to effectively support inter-flow coding.

We use the following network topologies: (i) canonical topologies shown in Figure 4.3, (ii) 5x5 grid topologies, (iii) 25-node random topologies, (iv) Roofnet topology with 35 nodes [121], (v) UW testbed topologies with 14 nodes [120]. Roofnet is an IEEE 802.11b testbed, whereas UW traces contain measurements from 802.11a and 802.11b testbeds. We also use both 802.11a and 802.11b in the synthetic topologies. Since the results under grid topologies are similar to the other topologies, they are omitted in the interest of brevity.

In 802.11a, each sender uses a transmission power of 10 dBm (Qualnet default) and a fixed PHY rate of 6Mbps, which gives $230m$ communication range and $1535m$ carrier sense range. In 802.11b, each sender uses transmission power of 15dBm (Qualnet default) and a fixed PHY rate of 2Mbps, which gives $1027m$ communication range and $3100m$ carrier sense range. We can certainly use another data rate for evaluation and expect similar relative performance. We compute the conflict graph by using these range values to determine if two links or nodes interfere.

Nodes are placed in a $1000m$ x $1000m$ area for 802.11a, and in a $2500m$ x $2500m$ area for 802.11b. In addition, we extend Qualnet to generate directional inherent packet losses. For the testbed topologies, the loss rates are based on the traces. For the synthetic topologies, the loss rates are uniformly distributed either between 0 and 30% (low loss), between 0 and 50% (medium loss), or between 0 and 80% (high loss).

We generate saturated UDP traffic with 1024-byte payload, and vary the number of flows from 1 to 16. Since the choice of routing protocols is important for multihop flows, our simulation randomly picks a source and destination that have at least 2 hops. For single-hop flows, all schemes with rate limiting can simply activate one-hop flows as much as possible and disable interfering multihop flows to achieve maximum throughput and the effects of routing cannot be not reflected. For each scenario, we conduct 10 random runs, each lasting 30 seconds. We report the average total throughput of these runs. In addition, the error bars on the graph show the standard deviation of the sample mean.

	O3	*O3*-Intra	MORE	COPE	SPP-RL	SPP
Linear chain	3.45	2.98	2.78	2.84	2.56	1.78
Diamond	1.50	1.11	0.91	0.47	0.47	0.40

Table 4.1: Total throughput (Mbps) for the topologies in Figure 4.3

4.6.2 Performance Results

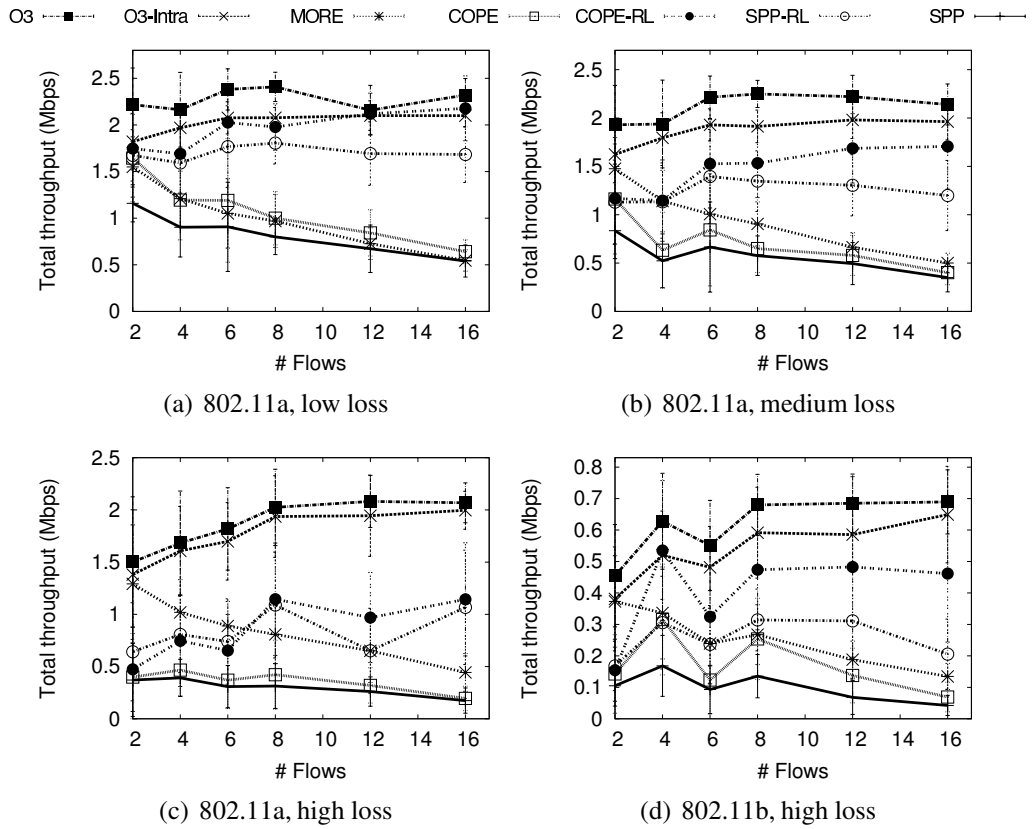
Canonical Topologies: Table 4.1 reports the throughput of the two canonical topologies shown in Figure 4.3. In the linear topology, there are two flows: from A to C and from C to A. Here, we observe that *O3* > *O3*-Intra > COPE > MORE > SPP-RL > SPP. SPP-RL outperforms SPP by 44% due to its proper rate limiting. COPE out-performs SPP by 60% due to inter-flow coding. *O3*, *O3*-Intra, and MORE out-perform SPP by taking advantage of opportunistic routing to effectively combat lossy wireless links. Among them, *O3*-Intra out-performs MORE through optimized rate limiting and opportunistic routing, while *O3* outperforms all the protocols by simultaneously exploiting inter-flow coding, opportunistic routing, and rate limiting. For the diamond topology with two flows, from A to D and from D to A, the relative ranking between various protocols remains almost the same, except a few differences. Here, COPE performs only slightly better than SPP and similarly to SPP-RL. This is because packet losses on the shortest paths significantly reduce the inter-flow coding opportunities. This also causes MORE to out-perform COPE by 93%. In contrast, *O3* can effectively take advantage of inter-flow coding over lossy wireless links and achieves the best performance. Its benefits over *O3*-Intra, MORE, COPE, SPP-RL and SPP are 35%, 65%, 219%, 219% and 275%, respectively.

Effects of number of flows in synthetic topologies: Figure 4.4 summarizes the performance results for 802.11a and 802.11b from low to high loss rates. We make the following observations.

First, *O3* out-performs state-of-the-art protocols in all the scenarios. For example, as shown in Figure 4.4(a), in low loss random topologies, compared with the protocols without rate limits, *O3* has 43-325% gain over MORE, 35-262% gain over COPE, and 92-329% gain over SPP; compared with the protocols with rate

limit, *O3* out-performs *O3*-Intra by 3-22%, COPE-RL by 2-29%, and SPP-RL by 32-38%.

The performance gain of *O3* comes from opportunistic routing, rate limiting, and inter-flow coding. In particular, we observe (i) *O3*, *O3*-Intra, and MORE out-perform SPP since opportunistic routing can more effectively cope with lossy wireless links, (ii) *O3* and *O3*-Intra out-perform MORE due to their optimized opportunistic routes and rate limits, and (iii) *O3* out-performs *O3*-Intra due to inter-flow coding. Note that the total throughput does not monotonically increase with the number of flows since we randomly select the flow sources and destinations and



(a) 802.11a, low loss

(b) 802.11a, medium loss

(c) 802.11a, high loss

(d) 802.11b, high loss

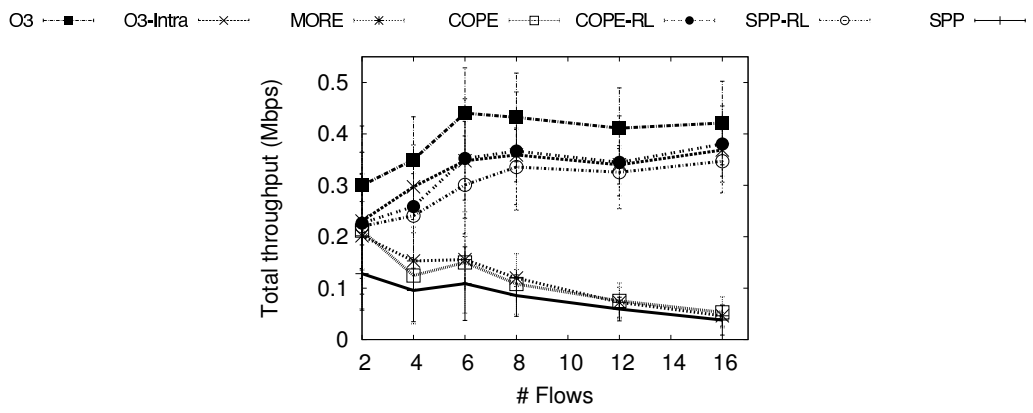
Figure 4.4: Total throughput in 25-node random topologies.

generate random link loss rates in each run.

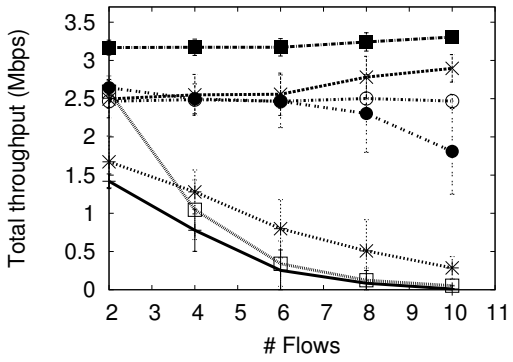
Second, rate limiting is important to all the protocols. In all cases, we observe the protocols with rate limiting significantly out-perform their counterparts without rate limiting. For example, as shown in Figure 4.4(a), *O3*-Intra out-performs MORE by 18-284%, COPE-RL out-performs COPE by 6-240%, and SPP-RL out-performs SPP by 44-211%.

Third, loss rate has significant impact on the effectiveness of opportunistic routing and inter-flow coding. In particular, as we would expect, the benefits of opportunistic routing increases with link loss rates. For example, comparing the results between low and high loss rates (Figure 4.4(a) and (c)), we observe that the gap between the performance gain of *O3* and *O3*-Intra over the other protocols increases. Moreover, MORE performs worse than COPE-RL and SPP-RL under low loss rate, and performs better than them under high loss rate because the benefit of opportunistic routing under high loss rate offsets the disadvantage arising from its lack of rate limiting. Moreover, the benefits of inter-flow coding decreases with link loss rates. For example, under high loss rate, *O3* has smaller gain over *O3*-Intra (3-9% gain), COPE performs similarly to SPP, and COPE with rate limiting performs similarly to SPP with rate limiting. Loss rates reduce inter-coding opportunities because when fewer packets are received at each node, they not only limit the choices of inter-flow coding and but also make the next hop harder to decode. Similar effects are observed in 802.11b as shown in Figure 4.4(d). Nevertheless, *O3* continues to out-perform the other protocols: it out-performs *O3*-Intra by 6-21%, COPE-RL by 17-194%, SPP-RL by 105-235%, MORE by 21-412%, COPE by 99-900%, and SPP by 273-1500%.

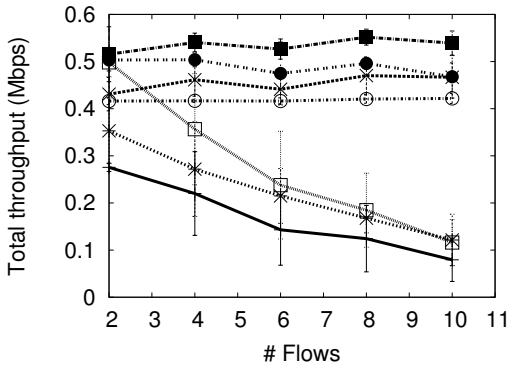
Effects of number of flows in testbed topologies: Figure 4.5(a), (b), and (c) show the performance results under Roofnet with 802.11b 1Mbps, UW testbed with



(a) Roofnet 802.11b



(b) UW 802.11a



(c) UW 802.11b

Figure 4.5: Total throughput in the testbed topologies.

802.11a 6Mbps, and UW testbed with 802.11b 1Mbps, respectively. In Roofnet, 68% of the links have within 1% loss and 80% of the links have within 57% loss. In UW 802.11a testbed, 75% of the links have within 1% loss and 80% of the links have within 51% loss. In UW 802.11b testbed, 52% of the links have within 1% loss and 80% of the links have within 93% loss. We make the following observations based on the performance results from these testbeds.

First, *O3* $>$ *O3*-Intra, COPE-RL, SPP-RL $>$ MORE, COPE $>$ SPP. The relative orderings of COPE-RL and *O3*-Intra depend on the loss rates: the former performs better under low loss and the latter is better under high loss.

Second, as in the synthetic topologies, all the protocols with rate limiting significantly out-performs their counterparts without rate limiting. For example, in Roofnet *O3*-Intra out-performs MORE by 15-696%, COPE-RL out-performs COPE by 1-617%, SPP-RL out-performs SPP by 71-811%.

Third, *O3* consistently out-performs all the other protocols. As shown in Figure 4.5(a), in Roofnet, *O3* out-performs *O3*-Intra by 14-30%, COPE-RL by 11-35%, SPP-RL by 21-46%, MORE by 48-810%, COPE by 41-694%, SPP by 134-

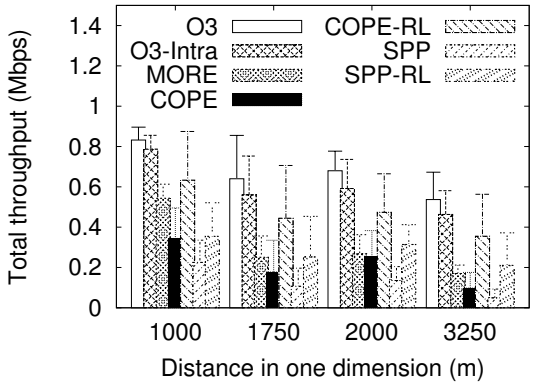
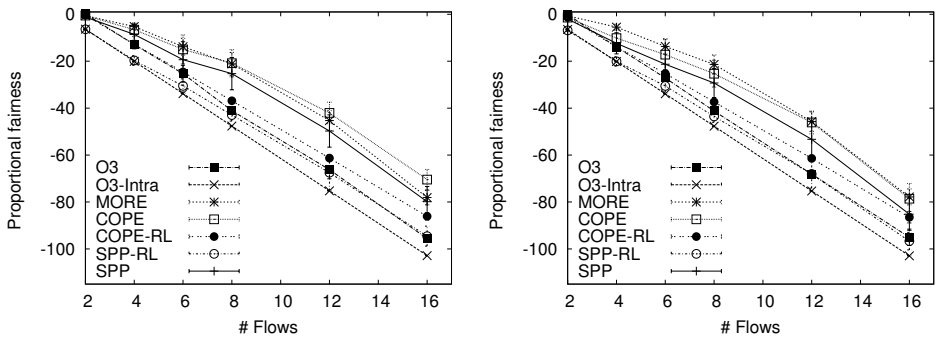


Figure 4.6: Throughput under varying network density in 25-node 802.11b random topologies (8 flows, high loss).

111%. As shown in Figure 4.5(b) and (c), in 802.11a and 802.11b UW testbed topologies, *O3* out-performs *O3*-Intra by 14-27%, COPE-RL by 2-83%, SPP-RL by 24-34%, MORE by 46-1000%, COPE by 3-6100%, SPP by 87-32600%.

Effects of network density: Next we vary the network density in 25-node 802.11b random topologies. We vary the area from 1000x1000m^2 to 3250x3250m^2. Figure 4.6 plots the total throughput. As we can see, *O3* out-performs the other protocols across all network densities. As before, rate limiting leads to significant performance improvement in all the routing protocols.

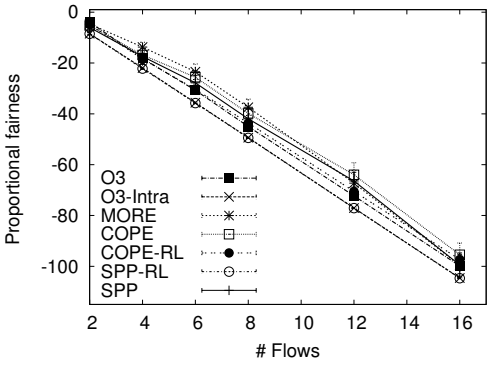
Proportional fairness under maximizing throughput: Finally, we evaluate pro-



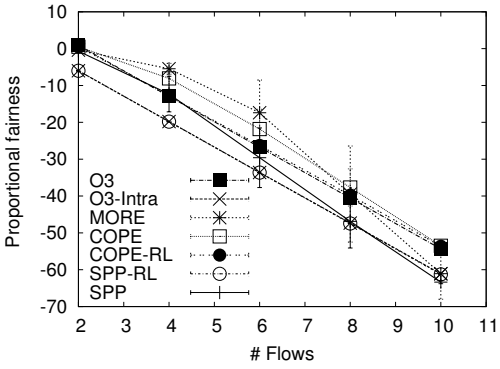
(a) 802.11a, low loss (b) 802.11a, medium loss

(c) 802.11a, high loss (d) 802.11b, high loss

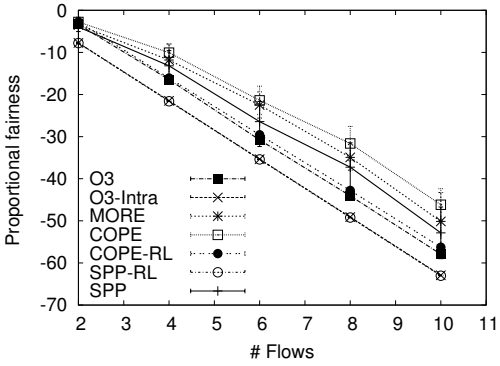
Figure 4.7: Proportional fairness while maximizing throughput using topologies in Figure 4.4.



(a) Roofnet 802.11b



(b) UW 802.11a



(c) UW 802.11b

Figure 4.8: Proportional fairness while maximizing throughput using testbed topologies in Figure 4.5.

portional fairness across the various protocols while the optimization objective is maximizing the total throughput. Proportional fairness [68] is defined as: $\sum_{f \in Flows} \log G(f)$, where $G(f)$ is flow f's throughput. Higher values are more desirable.

Figure 4.7 plots the proportional fairness using the synthetic topologies in Figure 4.4, and Figure 4.8 plots using the testbed topologies in Figure 4.5. We make the following observations. First, the protocols with rate limiting (SPP-RL, COPE-RL, O3-Intra, and O3) has worse proportional fairness than the protocols without rate limiting (SPP, COPE, MORE). Second, protocols using inter-flow coding has higher proportional fairness than those not using inter-flow coding. That is, among the rate-limited protocols, O3 and COPE-RL has higher proportional fairness than SPP-RL or O3-Intra, and among the protocols without rate-limiting, COPE has higher proportional fairness than MORE or SPP. This is because inter-coding by nature serves two or more flows, and thus optimizing the protocols with inter-coding will tend to activate more number of flows. These results suggest that solely maximizing throughput may lead to unfairness across different flows. Since our framework is general enough, when solving the linear program we can maximize not only the total throughput but also the propotional fairness, striking a good balance between the total throughput and the fairness.

4.7 Summary

Optimizing inter-flow network coding in opportunistic routing is useful but challenging due to the strong interactions between information splitting in opportunistic routing and inter-flow network coding. We approach the problem by proposing a novel hierarchical framework to decouple intra- and inter-flow network coding, and develop the first framework to jointly optimize opportunistic routing, rate

limiting, and network coding. We design a routing protocol to realize its benefit and demonstrate its effectiveness using Qualnet simulation. Furthermore, our simulation reveals the relative benefit of opportunistic routing, inter-flow coding, and rate limiting.

Chapter 5

VCD: Enabling High-bandwidth Vehicular Content Distribution via Opportunistic Communication

5.1 Introduction

In the previous chapters (chapter 3 and chapter 4), we study the crucial role of opportunistic communication in wireless mesh networks and present novel systematic optimization frameworks that give high end-to-end user performance. We also observe that cleverly applying inter-flow network coding into opportunistic routing can reap further performance benefit.

In this chapter, we explore opportunistic communication in vehicular networks. In addition to wireless mesh networks, vehicular networks have emerged from the strong desire to communicate on the move [12, 13, 43, 145]. Car manufacturers all over the world are developing industry standards and prototypes for vehicular networks (*e.g.*, [19, 25, 139]). In vehicular networks, the role of opportunistic communication becomes even more critical due to the following reasons. First, there is no persistent end-to-end path between a source and its destination. Second, as vehicles move at high speed, the topology of vehicular network changes rapidly, resulting in short contact time (only a few seconds), and such contacts are unplanned. Third, the movement of vehicles is constrained by the road and current traffic conditions, thus vehicles may make a sudden turn or stop, which makes it harder to predict the trajectory of vehicles. In this case, traditional routing or mobile routing schemes that assume end-to-end connection and simple movement of

nodes is not applicable. Leveraging opportunistic contacts among vehicles and APs are the only means to enable communication in vehicular networks.

Based on the observations above, we develop a novel optimization framework that leverages such opportunistic contact between vehicles and APs with the goal of high-bandwidth vehicular content distribution (VCD). To fully take advantage of the contacts between vehicles and APs, we proactively push content to the APs that the vehicles will likely visit in the near future. In this way, vehicles can enjoy the full wireless capacity instead of being bottlenecked by the Internet connectivity, which is either slow or even unavailable. Our VCD framework also cleverly employs both intra-flow and inter-flow network coding to efficiently store and distribute the content among the nodes in the network.

Challenges and opportunities: Cellular networks, despite good coverage, still have limited bandwidth and incur high cost. For example, many cellular service providers in US, like AT&T, T-mobile, Sprint, Verizon, charge around $60 per month for 5GB data transfer and $0.2/MB afterwards [98]. 5GB data transfer can only support 0.1Mbps for 111 hours (< 5 days)! The cellular service price in many other countries are similar or even higher [149]. Moreover, many mobile broadband providers restrict or limit large data exchanges, including streaming audio, video, P2P file sharing, JPEG uploads, VoIP and automated feeds [98]. According to the international poll of 2700 Devicescape customers [114], 81% smartphone users prefer Wi-Fi over 3G cellular for data services. Therefore there is strong need for supporting high-bandwidth applications in vehicular networks using Wi-Fi.

A natural way is to let a vehicle download content from the Internet when it meets an access point (AP) [13, 43]. However, it is challenging to meet high bandwidth requirement since vehicles often move at a high speed and thus the contact time between vehicles and APs tends to be short (*e.g.*, [26] reported that 70% of con-

nection opportunities are less than 10 seconds). In addition, it is often expensive to provide dense high-speed Internet coverage at a large scale. As a result, if vehicles fetch desired content on-demand from the Internet during their contact with an AP, the amount of data fetched may be insufficient to sustain the data rate required by applications such as video streaming when vehicles are outside the communication range of any APs.

With recent advances in wireless technology, Wi-Fi capacity has grown rapidly and can be at least an order of magnitude higher than typical Internet access link connectivity. For example, IEEE 802.11n can offer up to 600Mbps PHY data rate using 4 antennas. We performed a measurement experiment using a laptop equipped with NetGear WNDA3100 on a vehicle communicating with a NetGear WNDR3300 AP deployed near the road. We got 4.6Mbps using 802.11b, 22.2Mbps using 802.11g, and 39.7Mbps using 802.11n (2x2 MIMO) on 2.4GHz frequency, and 56.1Mbps using 802.11n on 5GHz. In comparison, DSL throughput ranges between 768Kbps to 6Mbps [8], which is an order of magnitude slower. The gap between the wireline and wireless capacity is likely to increase further (*e.g.*, due to the availability of new spectrum, such as whitespace, and advances in antenna and signal processing technology). Such large gap suggests that in order to enjoy high wireless capacity, we should proactively replicate content beforehand to the APs that a vehicle is likely to visit. While the idea of replication is natural, *how to replicate the content given the limited wireline and wireless resources and uncertainty in vehicular trajectory* is an open research question that we address.

Approach and contributions: We develop a replication strategy that effectively exploits the synergy among (i) Internet connectivity, which is persistent but has limited coverage and relatively low bandwidth, (ii) local wireless connectivity, which has high bandwidth but short contact duration, (iii) vehicle relay connectivity, which

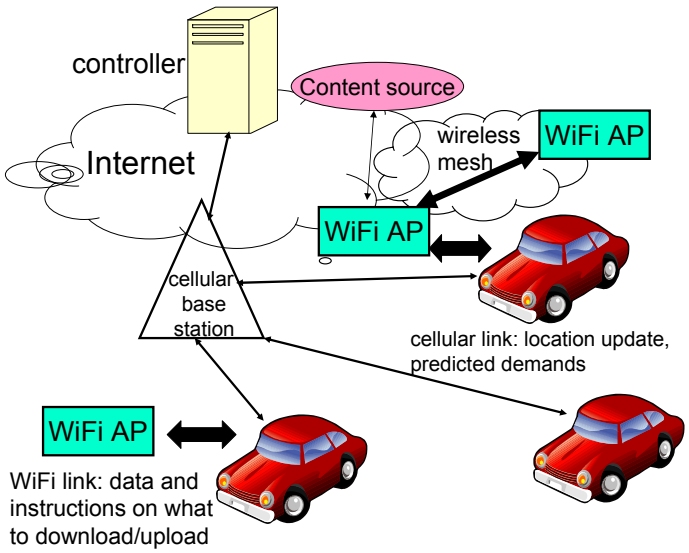


Figure 5.1: VCD architecture

has high bandwidth but high delay, and (iv) mesh connectivity among APs, which is persistent and has high bandwidth but low coverage. In particular, we optimize replication through wireline network and wireless mesh networks based on predicted mobility and traffic demands. Moreover, we opportunistically exploit the mobility of the vehicles to extend the coverage of the Internet and mesh connectivity. Even if only a small fraction of APs have Internet and mesh connectivity, by having the vehicles themselves relay content, one can potentially replicate content to a much larger number of APs. In essence, vehicle mobility has the potential to significantly increase the network capacity [52] and reduce future content access delay. Note that many mobile devices, such as smartphones, support the use of cheap external storage cards, which can help mitigate potential storage concerns regarding carrying traffic for other users in the system [136].

To this end, we develop a novel **V**ehicular **C**ontent **D**istribution (VCD) system for enabling high-bandwidth content distribution in vehicular networks. As

illustrated in Figure 5.1, VCD consists of vehicles, APs with and without Internet access (some of which may form a mesh network), content server on the Internet (*e.g.*, Web servers), and a controller. Vehicles submit location updates and content requests to the controller via cellular links. The controller optimizes the replication strategy based on predicted mobility and traffic demands, and instructs the APs to carry out the replication strategy. To enhance reliability and scalability, the controller can be replicated on multiple nodes. APs are deployed along road sides (*e.g.*, at gas stations and/or coffee shops) to allow vehicles on the road to opportunistically communicate with them. The APs prefetch content as instructed by the controller. Whenever a vehicle encounters an AP, the AP tries to send the requested content from its local storage if the content is available locally. Otherwise, the AP tries to fetch the content from an AP in the same mesh network if one is available. If no such AP is found, it fetches content from the Internet when it has Internet connectivity. In addition to sending the content that the vehicle itself needs, the AP may also send the vehicle content that can then be relayed to other APs, or fetch from the vehicle content that can be served to other passing vehicles later.

VCD systems are easy to deploy in practical settings. For example, a vehicular service provider (VSP) can install its own APs and/or subscribe to existing wireless hotspot services. Since it is easy to place a stand-alone AP than hooking it up with Internet connection, VCD is designed to explicitly take advantage of APs with and without Internet connectivity. An AP without Internet connectivity is still useful since it can serve as a static cache, which vehicles can upload content that can be served to other passing vehicles in the future.

VSPs can offer content distribution service to taxis, buses, subways, and personal vehicles. We focus on taxis and buses that offer high-bandwidth content distribution as a value added service to their passengers. These vehicles have low-

cost mobile devices on board for playing downloaded content. Such mobile devices can be installed by either the taxi/bus companies or VSPs. Since the mobile devices can be powered by the vehicles, power consumption is not an issue. The mobile devices interact with APs and the VCD controller to report required information (*e.g.*, location update and predicted traffic demands) and follow their instructions.

The key contributions of VCD include:

- *Optimized wireline and mesh replication.* To fully take advantage of short contact time between APs and vehicles, we replicate content in advance to the APs that will soon be visited by the vehicle. A distinctive feature of our replication scheme is that it is based on optimization. Specifically, we explicitly formulate a linear program (LP) to optimize the amount of data that can be delivered before the deadline under the predicted mobility pattern and traffic demands subject to given resource constraints (*e.g.*, short contact time and limited link capacity). The formulation involves addressing challenging modeling issues and is a valuable contribution. In contrast, previous works either focus exclusively on protocol-level optimization of one-hop communication between vehicles and APs (*e.g.*, [13, 24, 26, 102]), or rely on heuristics to guide data replication [29], or completely ignore the resource constraints [36], which are crucial in vehicular networks. Our formulation is highly flexible and can support both wireline replication (Section 5.2.2) and mesh replication (Section 5.2.3). The formulation can be efficiently solved using standard LP solvers (*e.g.*, *lp_solve* [86] and *cplex* [32]) owing to modern interior-point linear programming methods.
- *Opportunistic vehicular replication.* To further extend the coverage of the Internet and wireless mesh networks, we develop vehicular replication to opportunistically take advantage of local wireless connectivity and vehicular relay connectivity (Section 5.2.4). Different from traditional vehicle-to-vehicle (v2v) commu-

nication, our scheme leverages the APs as the rendezvous points for replicating content among vehicles since vehicle-to-AP communication is easier to deploy and such contacts are generally easier to predict than v2v contacts.

- *A new algorithm for mobility prediction.* For our replication optimization algorithms to be effective, it is critical to predict the set of APs a vehicle will visit in a future interval with high accuracy. Given the high driving speeds, diverse and unpredictable road conditions, infrequent location updates, and irregular update intervals, accurately predicting mobility is challenging in vehicular networks. We develop a new mobility prediction algorithm based on the idea of *voting among K nearest trajectories (KNT)* (Section 5.3). We also implement several state-of-the-art mobility prediction algorithms based on Markov mobility models [103, 133]. Our evaluation in Section 5.5 shows that *KNT* achieves better prediction accuracy on our dataset.
- *Thorough evaluation through simulation, emulation, and testbed experiments.* We conduct trace-driven simulations to evaluate the performance of VCD using San Francisco taxi [23] and Seattle bus traces [125] (Section 5.6). Our results show that VCD is capable of downloading 3-6X as much content as no replication, and 2-4X as much content as wireline or vehicular replication alone; mesh replication further helps to improve throughput by up to 22%. The benefit of VCD further increases as the gap between wireless and wireline capacity enlarges and the AP density increases. In addition, we have developed a full-fledged prototype VCD system that supports real video streaming applications running on smartphones and laptops (Section 5.4, 5.7 and 5.8). We deploy our system in two wireless testbeds using 802.11b and 802.11n. Live road tests suggest that our system is capable of providing video streaming to smartphone and laptop clients at a vehicular speed. To further evaluate the performance of VCD at scale, we run

the same AP and controller code as in the testbed together with emulated vehicles in the Emulab [42]. Our experiments show the efficiency of our implementation and that Emulab results closely follow the simulation results.

5.2 Optimizing Replication of VCD

In this section, we first present an overview of our system, and then develop wireline, mesh, and vehicular replication.

5.2.1 VCD Overview

At the beginning of every interval, the controller (shown in Figure 5.1) collects the inputs required for computing replication strategy. The controller computes the replication strategy during the current interval so that it can maximize user satisfaction during the next interval (Section 5.2.2). We use user satisfaction in the next interval as the objective since replication in the current interval is often too late to satisfy the traffic demands in the same interval. The controller then informs the APs of the replication strategy through the Internet or cellular network (in case the APs do not have Internet connectivity). We use cellular networks to send control messages as they are small. A vehicle performs the following actions during its contact with an AP:

- Step 1: The vehicle downloads the content according to the optimization results from this section.
- Step 2: After step 1, the vehicle may still have unsatisfied demand (*e.g.*, due to inaccurate prediction or insufficient capacity to replicate all the interesting content). The vehicle then downloads all the content that it is interested in and is also available locally at the AP.

- Step 3: Next, it downloads the remaining content that it is interested in from the AP's mesh network or the wireline network when the AP has wireline connectivity.
- Step 4: Parallel to the Internet download, the vehicle can take advantage of wireless capacity by opportunistically transferring files to and from APs (Section 5.2.4).

5.2.2 Optimized Wireline Replication

Problem formulation: Our goal is to find a replication strategy that maximizes user satisfaction subject to the available network capacity. Specifically, we want to determine how to replicate files to APs during the current interval to maximize the amount of useful content that can be downloaded by vehicles when vehicles meet the APs in the next interval. To support delay sensitive applications, only content that are downloaded before the deadline counts and the other content that already misses the deadline will be excluded from consideration for replication. This replication problem involves the following issues: (i) in what form to replicate the content, and (ii) how much to replicate for each file.

Applying network coding: To answer the first question, we note that directly replicating original content introduces two major problems. *First*, it is inefficient for serving multiple vehicles. Suppose multiple vehicles are interested in the same file and have downloaded different portions of the files before their contacts with an AP. If they visit the same AP, in order to satisfy all vehicles we need to replicate the union of the packets they need, which is inefficient. For example, vehicles 1 and 2 are both interested in file 1. Vehicle 1 has downloaded the first half and vehicle 2 has downloaded the second half before they encounter the AP. We need to replicate the complete file to satisfy both vehicles. *Second*, replicating original files is also

unreliable. Consider a vehicle is expected to visit three APs but in fact it only visits two of the three APs, which is quite common due to prediction errors. If we just split the file into three and transfer one part to each AP, then the vehicle will not get the complete file. However, if we split the files into two and transfer one part to each AP, the vehicle still may not get the complete file since it may get two redundant pieces (*e.g.*, when it visits the two APs that both have the first half of the file).

We apply network coding to solve both problems. Specifically, we divide the original content into one or multiple files, each containing multiple packets. We use random linear coding to generate random linear combinations of packets within a file. With a sufficiently large finite field, the likelihood of generating linearly independent packets is very high [57]. For a file with n packets, a vehicle can decode it as long as it receives n linearly independent packets for it.

Network coding solves redundancy problems in the multi-ple-vehicle case since each linearly independent packet adds value. In the above example of two vehicles, we only need to replicate one half worth of file content to satisfy both users, reducing bandwidth consumption by half. It solves reliability issue in the single vehicle case by incorporating redundancy. In the above example, we can split the file of interest into 2 and randomly generate 3 linear combinations of these 2 pieces and replicate one to each AP. Since any two pieces are linearly independent with a high probability, the vehicle can decode the file once it gets any two pieces.

Note that we need network coding (not just source coding) in order to avoid redundancy during replication without fine-grained coordination. That is, APs should re-encode data before they replicate data to vehicles and other APs. For example, AP 1 has a complete file 1, and sends to vehicle 1 half the file, which is relayed to AP2; similarly AP 1 sends half of the file 1 to vehicle 2, which relays it to AP2. In order to avoid replicating duplicates to AP 2, AP 1 should re-encode the

$\triangleright$ $Input:\ \ Intv, WCap, InCap, OutCap, CT, AP, size, has, Q$

$\triangleright$ $Output:\ \ x(f,i,a)\ and\ D(v,f,a)$

Maximize: $\sum_{v}\sum_{f}\sum_{a\in AP(v)} Q(v,f)D(v,f,a) - \gamma\sum_{i\in I}\sum_{a\in A}\sum_{f} x(f,i,a)$

Subject to:

[C1] $\sum_{f} D(v,f,a) \leq WCap(a) \times CT(a,v) \quad \forall v, a \in AP(v)$

[C2] $\sum_{a\in AP(v)} D(v,f,a) \leq size(f) - has(v,f) \quad \forall v, f$

[C3] $D(v,f,a) \leq has(a,f) + \sum_{s\in I} x(f,i,a) \quad \forall v, f, a \in AP(v)$

[C4] $\sum_{i\in I} x(f,i,a) \leq size(f) - has(a,f) \quad \forall f, a \in A$

[C5] $\sum_{i\in I}\sum_{f} x(f,i,a) \leq InCap(a) \times Intv \quad \forall a \in A$

[C6] $\sum_{a\in A}\sum_{f} x(f,i,a) \leq OutCap(i) \times Intv \quad \forall i \in I$

Figure 5.2: Optimizing wireline replication, where v is a vehicle, f is a file, a is an AP, i is a node with wireline connectivity (which may or may not be an AP, *e.g.*, a Web server), $Intv$ is an interval duration, A is the set of all the APs, I is the set of all the nodes with wireline connectivity, $AP(v)$ is the set of APs that vehicle v will visit, $Q(v,f)$ is the probability that v is interested in file f, $D(v,f,a)$ is the amount of traffic in file f vehicle v should download from AP a during a contact in the next interval, $x(f,n_1,n_2)$ is the amount of traffic in file f to replicate from node n_1 to node n_2 during the current interval, $CT(a,v)$ is average contact time of vehicle v at AP a, $WCap$ is wireless capacity, $InCap$ is incoming wireline access link capacity, $OutCap$ is outgoing wireline access link capacity, $has(n,f)$ is amount of file f a node n has, and $size(f)$ is the size of file f.

data before sending to the vehicles. In Section 5.4.2, we describe network coding cost and optimization.

Optimizing replication traffic: Using network coding, we transform the original problem of determining which packets to replicate into the problem of determining how much to replicate for each file. To solve the latter problem, we formulate a linear program, as shown in Figure 5.2. A few explanations follow. The first term in the objective function, $\sum_{v}\sum_{f}\sum_{a\in AP(v)} Q(v,f)D(v,f,a)$, quantifies user

satisfaction, which is essentially the total traffic downloaded by a vehicle (before the deadline), denoted as $D(v, f, a)$, weighted by the probability for vehicle v to be interested in file f, denoted by $Q(v, f)$. The second term in the objective represents the total amount of wireline replication traffic. We include both terms to reflect the goals to (i) maximize user satisfaction, and (ii) prefer the replication that uses less traffic among the replication strategies that support the same amount of traffic demands. Since the first objective is more important, we use a small weighting factor γ for the second term just for tie breaking (*i.e.*, when the first objective is the same, we prefer the one that has the lowest replication traffic). The value of γ should be small enough to ensure it does not dominate the first term, and our evaluation uses $\gamma = 0.001$. Note that in addition to optimizing the total downloaded traffic, it is also easy to support alternative metrics that are functions of downloaded traffic (*e.g.*, a linear approximation of proportional fairness, which balances between fairness and total downloaded traffic [119]).

Constraint C1 in Figure 5.2 enforces that the total amount of traffic downloaded from an AP during a contact is bounded by the product of AP's wireless capacity and average contact duration. Constraint C2 ensures that the total content downloaded for each file does not exceed the total file size minus the amount of file the vehicle already has before the download. Constraint C3 encodes the fact that the amount of file the vehicle can download from an AP cannot exceed what AP already has plus what will be replicated to the APs through the wireline network during the current interval. Constraint C4 indicates that the total replication traffic in file f towards an AP is bounded by the file size minus the amount that the AP already has. Constraints C5 and C6 reflect the total replication traffic through the wireline network does not exceed the access link capacity. The formulation can support APs with and without wireline access by setting wireline capacity to zeros

for APs without wireline access.

Obtaining input: As shown in Figure 5.2, we need $Intv$, $WCap$, $InCap$, $OutCap$, CT, AP, $size$, has, and Q. The $Intv$ is a control parameter that determines how frequently the optimization is performed. In our evaluation, we set $Intv$ to be 3 minutes, which gives a good balance between (i) achieving accurate mobility prediction and (ii) limiting the optimization overhead, since both (i) and (ii) decrease as $Intv$ increases. The next three inputs on link capacity—$WCap$, $InCap$, and $OutCap$—are known in advance and change infrequently. CT is estimated using historical data and only needs to be updated infrequently. For ease of estimation, in our evaluation we set $CT(a, v)$ to be the average duration of all contacts from the trace. AP can be obtained by either letting a vehicle run a mobility prediction algorithm locally or have it send several of its recent GPS coordinates to the controller, which will perform mobility prediction. $size$, has, and Q are reported by the vehicles either through a Wi-Fi link during a contact with an AP or via a cellular link during other time. A vehicle predicts what future content to request based on the previous and current requests. For streaming content, it is relatively easy to predict as most users will request the subsequent frames. Demand prediction in general has been a well-researched problem in many domains [12, 109] and we can leverage existing solutions. Note that all the control information is small and can be easily compressed by sending delta from the previous update.

Using optimization results: *To enhance robustness against errors in estimating the inputs, we use $x(f, i, a)$ and $D(v, f, a)$ to control the relative replication and download rates across different files using the weighted round robin scheduling.* For example, if $x(f1, i, a) = 2 * x(f2, i, a)$, file 1 is downloaded twice as fast as file 2. In this way, we can still fully utilize network resources even if contact time or network capacity have estimation errors.

5.2.3 Optimized Mesh Replication

If some APs along the road are close together, they can form a mesh network. The mesh connectivity indicates that (i) we can now replicate content to the APs using mesh connectivity in addition to wireline connectivity, and (ii) if a vehicle meeting AP1 requests a file that AP1 does not have, it is more efficient to fetch from its mesh network (if there is an AP having the file) than fetching via the slow wireline access link. A neighboring AP in the mesh network can have the file either due to explicit replication or opportunistically caching from earlier interactions.

To support (i), we make the following modifications to the replication formulation in Figure 5.2. Let $MCap(a', a)$ denote the capacity of a wireless link from AP a' to a in the mesh network, which can be different from the capacity of wireless links between vehicles and APs ($WCap$). Let $z(f, a', a)$ denote the amount of content to replicate from AP a' to a for file f through the mesh network. Let $ETX(a', a)$ denote the average number of transmissions required to send a packet from a' to a through the mesh and can be easily estimated by measuring link loss rate using broadcast probes as in [31]. Our modifications include (1) adding $-\gamma \sum_f \sum_{(a',a) \in mesh} z(f, a', a)$ to the objective function to prefer the replication that uses less wireline and mesh replication traffic among the ones that support the same traffic demands, (2) adding $+ \sum_{(a',a) \in\ mesh} z(f, a', a)$ to the right handside of [C3] to indicate a node can download from AP a any content that is already available at a or replicated to a through either the wireline or mesh network, (3) adding the following two new constraints: $z(f, a', a) \leq has(a', f)$ and $\sum_{f,(a',a) \in\ mesh} \frac{ETX(a',a) z(f,a',a)}{MCap(a',a)} \leq 1$. The former constraint ensures AP a' cannot replicate more content than it has. The latter is interference constraint, which enforces that total active time of all mesh nodes cannot exceed 100% assuming all nodes in the mesh network interfere with each other. Note that its left-hand side

computes activity time by multiplying the replicated content by the expected number of transmissions normalized by the wireless capacity.

To support (ii), when AP a receiving a request for a file that it does not have locally, it first tries to get from AP a' in the same mesh if the end-to-end throughput (approximated as $MCap(a', a)/ETX(a', a)$) is higher than the wireline access link; only when no such AP is found, does it fetch using the wireline access link.

5.2.4 Opportunistic Vehicular Replication

In addition to wireline and mesh replication, content can also be replicated using vehicles – a vehicle can carry content from one AP to another as it moves. This new form of replication is more effective than traditional vehicle-to-vehicle (V2V) replication, because V2V needs a very large number of vehicles to be effective whereas even a small number of APs can significantly enhance the performance by leveraging the Internet and mesh connectivity among them [15].

One way to support this new vehicular replication is to augment the LP formulation in Figure 5.2 with vehicular replication terms, which can produce wireline, mesh and vehicular replication as the final output. However, due to unpredictability in vehicular relay opportunity, we find the effectiveness of such optimization is rather limited. Interestingly, we find the following simple opportunistic vehicular replication scheme is effective.

Since the wireline fetch is bottlenecked by the slow access link, the wireless link is not fully utilized. Therefore, as mentioned in Section 5.2.1, parallel to the wireline fetch, a vehicle can take advantage of local wireless connectivity to exchange content with the AP. Such exchange has two benefits: (i) the vehicle can upload content to the AP, which can serve other vehicles later, and (ii) the vehicle

can download files, which may serve its own demand in the future or the vehicle can relay the content to other APs for future service. To enhance effectiveness, we order the files to upload based on the expected future demand for the file at the AP, which is estimated as $\sum_{v:\ v \text{ visits } a} Q(v, f) demand(v, f)$, where $demand(v, f)$ is the expected size of file f vehicle v is interested in. While this vehicular replication is simple, our evaluation shows that it is highly effective.

5.3 Predicting Mobility

If we can predict the AP that a vehicle will visit, we can start replicating the required content to the AP well before the vehicle arrives so that the vehicle can enjoy high wireless bandwidth during its download. Predicting mobility for vehicles is challenging because (i) vehicles often move at high speed, which implies that there can be many possible next states and it is difficult to accurately predict transitions to a large number of next states, (ii) the GPS updates often have relatively low frequency (*e.g.*, once per minute) and tend to arrive at irregular intervals, and (iii) the road and traffic conditions are highly dynamic and difficult to predict.

To address the challenge, we develop a novel mobility prediction algorithm for vehicular networks: *K Nearest Trajectories (KNT)*. We also implement two existing algorithms based on Markov mobility models [103, 133]. In Section 5.5, we show that *KNT* achieves better accuracy on our dataset.

Algorithm: We observe that the mobility of vehicles exhibits unique structure – a vehicle follows the roads and only makes turns at the street corners or highway exits. This suggests that a good predictor should take into account the speed and direction in the previous interval as well as the underlying road structure. Our *KNT* algorithm is able to account for such information without requiring explicit

knowledge about the detailed road map. Given a vehicle v_0 and current time t_0, the algorithm predicts the set of APs visited by v_0 in a future interval $[t_0 + \Delta_1, t_0 + \Delta_2]$ ($\Delta_2 \geq \Delta_1 \geq 0$) in two steps:

1. *Finding K nearest trajectories.* Our algorithm first finds K existing mobility trajectories in a GPS location database that best match the recent mobility history of the given vehicle. Specifically, we maintain a database of past GPS coordinate updates: $\mathcal{D} = \{(v, t, c)\}$, where v is a vehicle, t is the time for the update, and c is the GPS coordinate. For any vehicle v and current time t, we define its mobility history MH as the set of GPS coordinates reported by v in the past δ seconds: $MH_v^t = \{c | (v, s, c) \in \mathcal{D} \wedge s \in [t - \delta, t]\}$. We also define a distance function between two trajectories: $f(MH_{v_0}^{t_0}, MH_v^t) = \sum_{c \in MH_{v_0}^{t_0}} \min_{d \in MH_v^t} \|c - d\|_2$, where $\|c - d\|_2$ is the Euclidean distance between the two locations specified by GPS coordinates c and d. Essentially, this distance function reflects the total distance from each point on $MH_{v_0}^{t_0}$ to the closest point on MH_v^t. We then find K pairs of (v, t) that minimizes $f(MH_{v_0}^{t_0}, MH_v^t)$, *i.e.*, the K nearest neighbors of (v_0, t_0).
2. *Voting.* For each of K nearest trajectories (v, t), we use linear interpolation (*i.e.*, using a line to connect two adjacent points) to obtain its mobility trajectory in the future interval $[t + \Delta_1, t + \Delta_2]$. Based on this, we obtain the set of APs visited by v during that interval. We then report all the APs that are visited by at least T out of K nearest trajectories as the predicted set of APs that will be visited by v_0 during future interval $[t_0 + \Delta_1, t_0 + \Delta_2]$.

In step 1 above, to avoid computing $f(MH_{v_0}^{t_0}, MH_v^t)$ for all pairs of trajectories (which is expensive), we only compute for the trajectory pairs that are nearby. To quickly identify the trajectories that are close to the current one, we create an efficient index structure by (i) discretizing the GPS latitude-longitude coordinate

space into $0.0001^\circ \times 0.0001^\circ$ grid squares, and (ii) storing all the (v, t) inside each grid square. Given (v_0, t_0), we start from its grid square and use expanded ring search to find C candidate points (v, t) residing in the same or nearby grid squares. We then find K nearest neighbors among these C candidate points.

To be general, our prediction algorithm intentionally does not exploit external knowledge (*e.g.*, certain vehicles have similar trajectory on different days, which may hold for some personal vehicles). When such information is available, our prediction algorithm can potentially incorporate it when finding nearest trajectories to further improve the accuracy.

Parameter setting: Our algorithm has four control parameters: the number of nearest trajectories K, the number of candidate points C, the voting threshold T, and the mobility history duration H. In our evaluation, we keep $C = 32$, vary $T = 1, 2$, vary K from 2 to 12, and vary H from 60 to 180 seconds. Our results show that $(K = 4, T = 2, C = 32, H = 60)$ consistently give the best performance. We thus only report the results under this parameter setting.

5.4 VCD Implementation

We implement VCD in both Emulab [42] and our real testbed with smartphone and laptop clients. VCD consists of a controller, APs, content servers, and clients in vehicles. Emulab and testbed use the same controller, AP, and content server implementation, all of which are implemented as multi-threaded C++/Linux programs. They differ in client implementation. In Emulab, we implement a virtual vehicle program, which can emulate multiple vehicles, allowing us to conduct a trace driven emulation of all the vehicles in our trace using a few virtual vehicles. The client in the real testbed is implemented on both smartphones and laptops,

which is described in Section 5.4.2.

5.4.1 System Overview

Communication between APs and controller: The APs and controller communicate with each other using TCP. As noted in Section 5.2.1, at the beginning of every interval the controller collects inputs, computes the replication strategy, and instructs content servers or APs to perform wireline and mesh replication at the desirable rates.

Communication between AP and vehicle: The communication between APs and vehicles uses UDP that sends data at close to the PHY data rate. When a vehicle contacts an AP, it sends a HELLO message that includes (i) a list of files and their sizes that it already has, (ii) the files it is interested in during the current and next intervals. Upon receiving the first HELLO message from the vehicle, the AP initiates data download to the vehicle according to the four steps described in Section 5.2.1. Meanwhile, the vehicle also sends buffered GPS updates (generated every 20 seconds in the testbed and every 1 minute in Emulab). In step 4, the AP determines a list of files for the vehicle to upload sorted in increasing utility as described in Section 5.2.4. The AP sends this list in a REQ message. Upon receiving the first REQ message, the vehicle initiates data upload to the AP. Both HELLO and REQ messages use soft state and are sent periodically once every control interval (100ms in testbed and 1s in Emulab). These messages also serve as heartbeats to the other party.

To achieve efficiency and reliability for data traffic, an AP applies network coding before sending the data it receives. In addition, we use multiple content servers and leverage a central dispatcher to distribute requests to an appropriate content server for load balancing.

5.4.2 Client Implementation

We implemented client on both Windows XP laptops and smartphones. We use HP Ipaq 910 Business Manager smartphones with Windows Mobile 6.1 Professional operating system, Marvell PXA270 416 MHz Processor, 128MB RAM, Marvell SDIO8661 802.11 b/g Wi-Fi card, and the .Net Compact Framework. Our implementation on smartphones uses OpenNet API, and that on Windows uses Managed Wi-Fi API. Implementing on smartphones introduces several challenges: (i) limited APIs and often inconsistent implementations, (ii) expensive I/O, (iii) limited system resources, and (iv) many existing wireless optimizations cannot be implemented due to lack of low level access, which we address.

Handling expensive I/O: Since I/O on smartphones is around an order of magnitude slower than desktops, packets cannot be stored on the disk and read back on-demand for vehicular replication. For simplicity, we use an in-memory packet buffer with FIFO replacement policy. We further limit disk access during the contact with APs and push data to the disk only after the contact is over so that we can fully utilize the short contact time for data transfer.

Handling network coding cost: Due to the slow processor, thread scheduling and dynamic assignment of priorities are important. For example, network coding incurs much higher cost on the smartphone than on the desktop as shown in Table 5.1. We use packet size of 1230 bytes (*i.e.*, the packet payload in our testbed implementation to ensure the maximum packet size is still within 1500 bytes (Ethernet MTU)). Our evaluation uses file sizes of 35, 70, and 110 packets, which correspond to minimum, median and maximum file sizes used in our experiments. To minimize the impact of decoding, we schedule the decoding thread at a low priority during a contact and increase its priority after the contact.

Batch size	110 packets		70 packets		35 packets	
Device	Phone	Desktop	Phone	Desktop	Phone	Desktop
Encoding	12.19s	0.0228s	4.79s	0.0088s	1.18s	0.0021s
Decoding	8.22s	0.017s	3.27s	0.0067s	0.809s	0.0012s

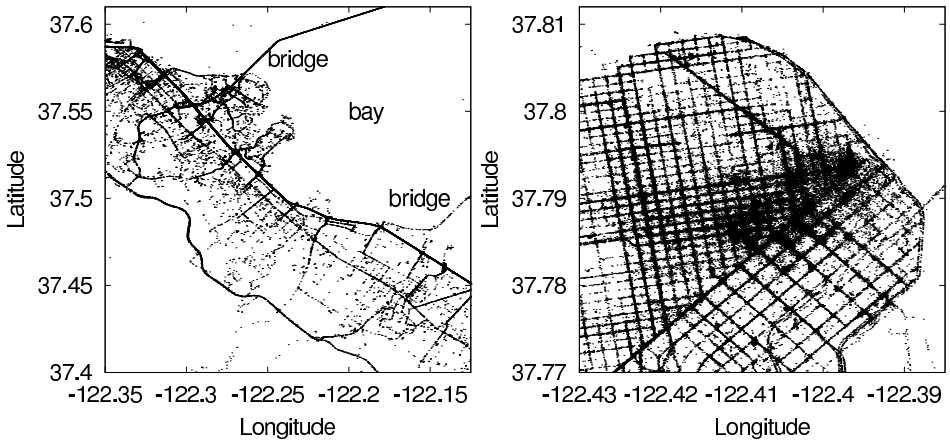
Table 5.1: Network coding benchmarks

Connection setup: The ability to quickly establish connection to an AP is crucial. [22, 54] examine this problem in greater detail. In the context of smartphones, the problem becomes even harder since NDIS does not provide access to many low level parameters to implement the association optimizations proposed in the literature. Windows Mobile provides two ways to initiate connection to a Wi-Fi network programmatically, either through the wireless zero config (WZC) interface or by setting the appropriate NDIS OIDs. The association times using the WZC interfaces were around 3.0 sec, which is unacceptable in the vehicular network context. We therefore disable WZC and implement NDIS based association, which yields significantly lower association times. We also implement our own DHCP client and use the DHCP caching mechanism described in [22].

Our connection setup procedure is as follows. The smartphone scans for APs every 100 ms. When an AP is discovered, the smartphone waits for 3 RSSI readings greater than -91dB before trying to associate. We do not associate immediately because an association failure is expensive. The association procedure is retried up to 7 times with a short delay of 50ms between consecutive attempts. The various threshold values used in the scheme were chosen empirically. We report the association time and failures in Section 5.8.

5.5 Mobility Prediction Accuracy

Mobility traces: We obtain real vehicular mobility traces from Cabspotting [23] and Seattle [125]. The former contain over 10 million GPS longitude and latitude



(a) Taxi locations (on highways) (b) Taxi locations (inside city)

Figure 5.3: Illustration of traces for mobility prediction.

coordinates for approximately 500 taxis in the San Francisco Bay Area over the course of 30 days (December 13, 2008 – January 13, 2009). The latter contains several week-long traces of city buses in Seattle during 2001. The bus system consisted of over 1200 vehicles covering a 5100 square kilometer area. The GPS coordinates are updated approximately once per minute for both Cabspotting and Seattle traces. Figure 5.3 (a) and (b) illustrate the vehicle locations along the highway and inside San Francisco. One can clearly observe the underlying street structure from taxis' GPS. Similar pattern was observed in Seattle traces.

AP locations: We consider two sets of locations for placing APs: (i) gas stations and (ii) coffee shops. We use Yahoo's Local Search API (version 3) [150] to obtain the longitude and latitude coordinates of 1120 gas stations and 1620 coffee shops in San Francisco Bay Area, as well as 618 gas stations and 738 coffee shops in Seattle. The average distance between two closest APs in the traces ranges between $345 - 589\ m$ and the median distance is $157 - 433\ m$. There are quite a few APs whose distance exceeds $3500\ m$ in all the four traces. The communication range between an AP and a vehicle is set to either 100 or 200 meters. We use these val-

ues because they approximate the communication ranges we measured from our vehicular testbeds using 802.11b and 802.11g, respectively. To determine the contact period between a vehicle and an AP, we use linear interpolation to obtain the vehicle's mobility trajectory between two adjacent GPS location updates.

Trace statistics: We analyze the traces and find that 23% – 40% of time the vehicles were parked or moved within 1 mile/hour, 70% of time they moved less than 11 – 15 miles/hour, and 90% of time they moved less than 25-27 miles/hour. Since most of the cabs are in the downtown area, they are bounded by the speed limits of the downtown area. We further study the contact duration and observe 70% of the contacts between a vehicle and an AP last less than 39-51 seconds when the communication range is 100 meters, and less than 54-82 seconds when the range increases to 200 meters. Such short contacts highlight the importance of replicating data in advance.

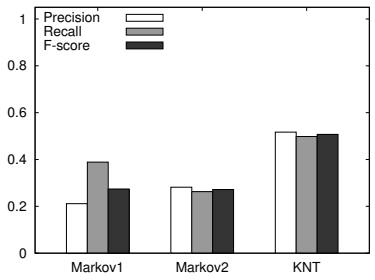
Baseline algorithms: For baseline comparison, we implement a variant of the mobility prediction algorithm in [103]. The algorithm is based on a second-order Markov mobility model. Each state has two sets of coordinates: the vehicle's location at time τ ago, and its current location. In our evaluation, τ is either 1 or 2 or 3 minutes. We deal with irregular GPS update intervals through linear interpolation. To avoid state space explosion, the algorithm discretizes the longitude and latitude coordinates into $0.001^\circ \times 0.001^\circ$ grid squares. The algorithm uses past mobility traces to learn the probability for a vehicle to transition into any new grid square given its last and current grid squares. Based on the transition probabilities, the algorithm identifies the grid square that the vehicle is most likely to visit next, and uses the center of this grid square as the predicted new location for vehicle after time τ. This procedure is repeated to make predictions further into the future. Based on the predicted locations, the algorithm applies linear interpolation to ob-

tain the entire mobility trajectory and then computes the set of APs the vehicle is predicted to visit during a future interval. As in [103, 133], the algorithm falls back to a first-order Markov model when the second-order Markov model fails to make a prediction. Finally, we also implement the first-order Markov model as another baseline algorithm.

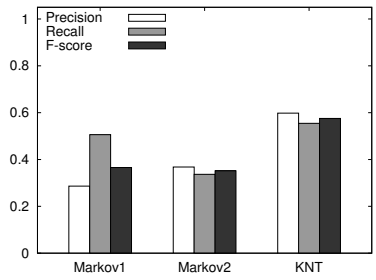
Metrics: We quantify the prediction accuracy using two metrics: (i) *precision*, *i.e.*, the fraction of APs predicted by our algorithms are indeed visited by the vehicles in a future interval, and (ii) *recall*, *i.e.*, the fraction of APs visited by the vehicles in a future interval are correctly predicted by our algorithms. In addition, we integrate precision and recall into a single metric called *F-score* [148], which is the harmonic mean of precision and recall: $\textit{F-score} = \frac{2}{1/precision+1/recall}$. For all three metrics, larger values indicate higher accuracy.

Evaluation results: We consider the following prediction scenario as required by our replication optimization algorithm: *per-interval prediction*, which divides time into fixed intervals and the goal is to predict the set of APs that will be visited by a vehicle in the next interval. The prediction interval is set to 3 minutes, which matches the interval for periodic replication optimization. For each prediction algorithm we evaluate, we consider multiple parameter configurations and choose the configuration that yields the best *F-score*. The results from Cabspotting traces use seven days of training data to predict the mobility on the eighth day, and results from Seattle bus traces use 5 days of training data to predict the sixth day as these traces have shorter duration.

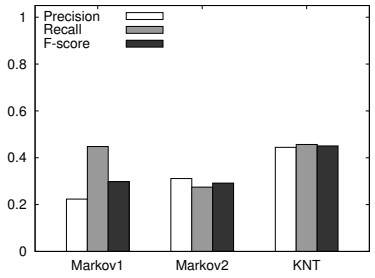
Figure 5.4 shows the prediction accuracy when APs are placed at either gas stations or coffee shops and the communication range is either $100m$ or $200m$. For the San Francisco taxi mobility trace (Figure 5.4 (a)–(d)), the *F-scores* of our algorithm (*KNT*) are 25-85% higher than those of the first-order Markov model



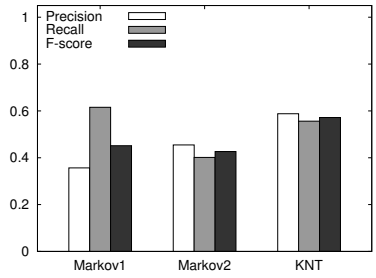
(a) San Francisco, gas, range=100m



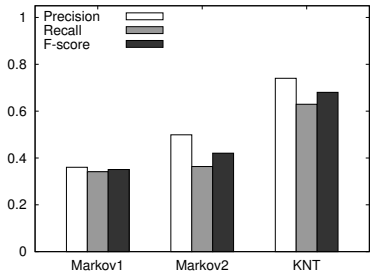
(b) San Francisco, gas, range=200m



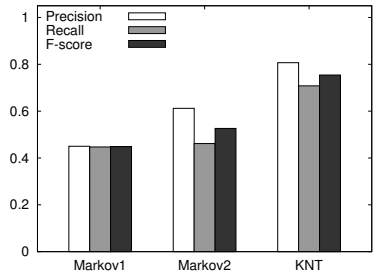
(c) San Francisco, coffee, range=100m



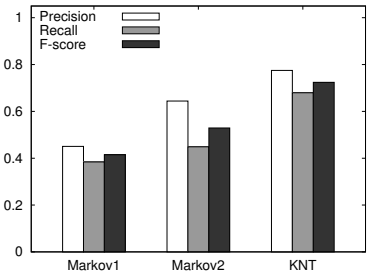
(d) San Francisco, coffee, range=200m



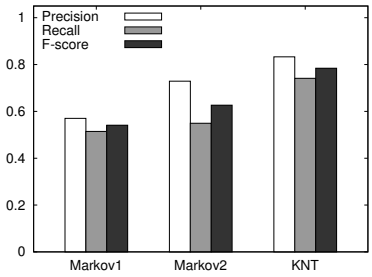
(e) Seattle, gas, range=100m



(f) Seattle, gas, range=200m



(g) Seattle, coffee, range=100m



(h) Seattle, coffee, range=200m

Figure 5.4: Accuracy comparison of different mobility prediction algorithms.

(*Markov1*) and second-order Markov model (*Markov2*). For the Seattle bus mobility trace (Figure 5.4 (e)–(h)), *KNT* outperforms *Markov1* and *Markov2* by 25–94% in terms of *F-scores*. In general, the absolute prediction accuracy for all three algorithms is higher for the bus mobility trace, because buses tend to follow fixed routes and are thus more predictable.

Finally, it is worth noting that in contrast to findings in [103, 133], *Markov2* does not significantly outperform *Markov1* in our evaluation. This suggests that with higher speed and less frequent GPS location updates, mobility prediction is more challenging in vehicular networks. As a result, solutions that perform better in less mobile environment do not necessarily perform better in vehicular networks.

Summary: The above results clearly show that our *KNT* mobility prediction algorithm consistently achieves good accuracy in vehicular networks. Later in Section 5.6, we further show that optimization based on our prediction results yields good performance in practice.

5.6 Trace-Driven Simulation

5.6.1 Simulation Methodology

We develop a trace-driven simulator for evaluation as follows. We first generate the contact traces based on the mobility traces, AP locations, and wireless communication range. When multiple vehicles contact an AP at the same time, we divide the original contacts into non-overlapping contacts, each of which has only one vehicle in contact with an AP. Such contact partitions can be easily realized in practice by letting the AP serve the new vehicle only after it finishes serving the previous one. Similarly, when a vehicle is within the communication range of multiple APs, we also partition the contact into multiple non-overlapping intervals,

each of which involves one AP. Another way to partition a contact between multiple vehicles and an AP or between multiple APs and a vehicle is to equally divide the contact time among multiple vehicles or multiple APs that are involved in the contact to mimic round-robin scheduling. The performance of these two types of partitions is similar, and we use the first partition in our evaluation.

We then feed the actual contact traces (after the above post processing), predicted contacts, and traffic demands to the simulator. The simulator updates the content at APs and vehicles based on the actual contacts, traffic demands, replication schemes, and wireless and wireline capacity at APs. We implement network coding for all data transfer to ensure only innovative packets (*i.e.*, whose coding coefficients are linearly independent) are exchanged between APs and vehicles or among APs. We have a content server on the Internet, which has all the content, whereas all APs and vehicles are initialized with no content.

We compare (i) no replication, (ii) wireline replication alone, (iii) vehicular replication alone, (iv) both wireline and vehicular replication, (v) wireline, vehicular, and mesh replication (VCD). In all the schemes, a vehicle downloads content remotely from the Internet whenever the AP has Internet connectivity and the content is not available locally at the AP or mesh network.

To study the impact of traffic demands, we generate traffic demands following either uniform or Zipf-like distribution. In both cases, for every interval, a vehicle randomly selects a specified number of files to request. In the uniform distribution, a file is uniformly drawn from the pool of the files that the vehicle has not requested previously. In Zipf-like distribution, the probability of requesting the ith file is proportional to $\frac{1}{i^\alpha}$, where i is the popularity ranking of the file and $i = 1$ indicates the most popular file. We set $\alpha = 0.4$ so that we can generate similar traffic load using both Zipf-like and uniform distributions and the performance difference

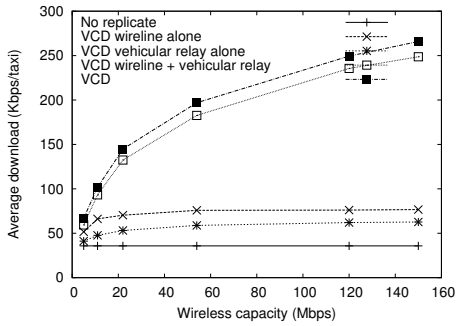
is solely due to the difference in the distribution.

For delay sensitive applications, such as video, their performance depends on the amount of data received before the deadline. Therefore, we use average throughput per vehicle as our performance metric, which denotes the total demand that is satisfied before the deadline divided by the product of the number of vehicles and the entire trace duration (including the time without contacts with APs). The deadline is set to the end of the interval in which the demand is generated.

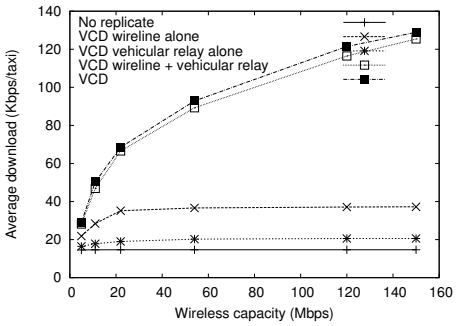
Our evaluation uses 2-hour trace, which exhibits similar contact characteristics as in the 1-day trace, shown in Section 5.5. Other default settings used in our evaluation include: 100-meter communication range between APs and vehicles, 500-meter communication range among APs (well within reach by many mesh routers [9,93]), Zipf-like traffic demands, placing APs at coffee shops, all APs having 22 Mbps wireless link, half of the APs having Internet links with 2Mbps while the other half have no Internet connection. The content server has a 1 Gbps Internet link and zero wireless capacity to indicate that it is not directly reachable by vehicles. There are 1200 files in total. Each user requests 20 files every 3-minute interval, each file has 2K packets, which contains 1000 bytes. Every file represents either a video clip or one chunk in a larger video file (*e.g.*, We divide a large video file into smaller chunks and generate random linear combinations of packets within each chunk for efficient replication). We further evaluate the effects of changing these parameters.

5.6.2 Simulation Results

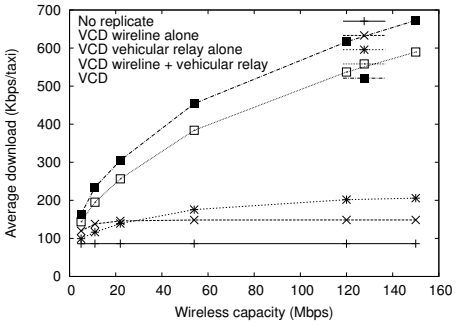
Varying wireless bandwidth: In Figure 5.5, we plot the total downloaded content as we vary wireless bandwidth from 5, 11, 22, 54, 120, and 150 Mbps. We make the following observations. First, in all cases VCD significantly out-performs



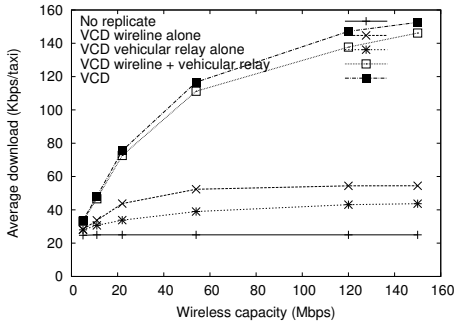
(a) San Francisco, coffee shops, range=100m



(b) San Francisco, **gas station**, range=100m



(c) San Francisco, coffee shops, **range=200m**



(d) **Seattle**, coffee shops, range=100m

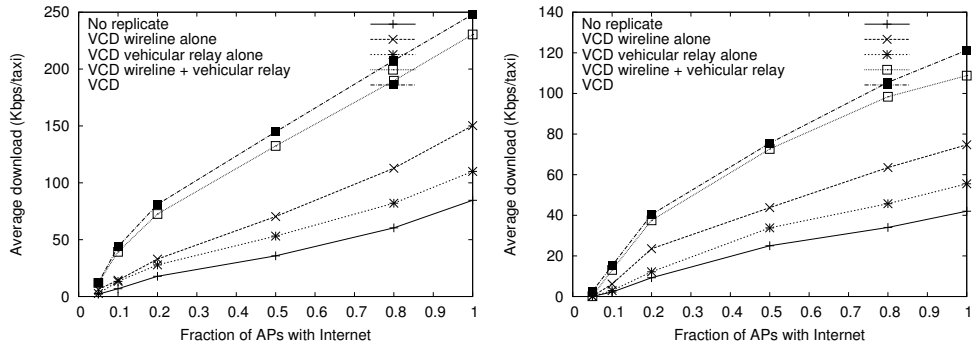
Figure 5.5: Average throughput of 50 vehicles under varying wireless capacity and Zipf-like traffic demands. The difference from the base configuration is in bold.

the other schemes and its benefit increases rapidly with wireless capacity. Second, as we would expect, no replication performs the worst. Interestingly, its performance remains the same as we increase wireless capacity. This is because without replication APs often do not have content locally and the wireless download is bottlenecked by slow Internet access capacity. This further demonstrates the need of replication. Third, the performance of both wireline and vehicular replication alone initially improves with increasing wireless capacity and then tapers off. This is because limited Internet capacity prevents fully taking advantage of large wireless capacity. In comparison, harnessing both wireline and vehicular replication oppor-

tunities can effectively utilize the large wireless capacity when available. Adding mesh replication further increases average throughput by 14-20% under high AP density (Figure 5.5(c)), and by 3-13% in low AP density. The benefit of mesh replication can be increased further if APs use high gain antennas or MIMO. Overall, at 22Mbps Wi-Fi capacity, VCD achieves 70 – 300 Kbps average throughput per vehicle depending on the AP density, which can support video streaming applications.

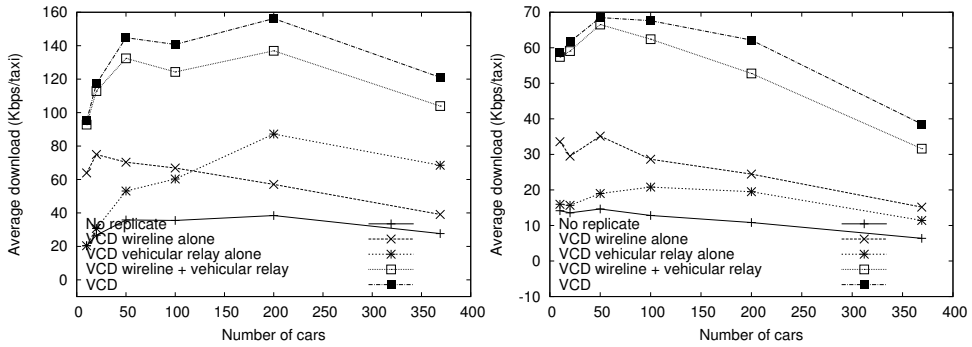
Varying fraction of APs with Internet connectivity: Next we vary the fraction of APs with Internet connectivity. Figure 5.6(a) and (b) plot the average downloaded traffic in San Francisco and Seattle traces, respectively. As we can see, VCD continues to significantly out-perform the other schemes. In addition, the benefits of all types of replication increase with the fraction of APs that have Internet connectivity. The rate of such increase is faster for the replication schemes that involve wireline replication, since they explicitly take advantage of the new wireline capacity to push data.

Varying number of vehicles: To further evaluate the impact of degree of deployment, we vary the number of vehicles by randomly selecting a subset of vehicles



(a) San Francisco (b) Seattle

Figure 5.6: Average throughput under varying fraction of APs with Internet (Zipf-like traffic, APs at coffee shops, range=100, 50 vehicles).



(a) coffee shops (b) gas stations

Figure 5.7: Average throughput under a varying number of vehicles (San Francisco, Zipf-like traffic, range = 100m).

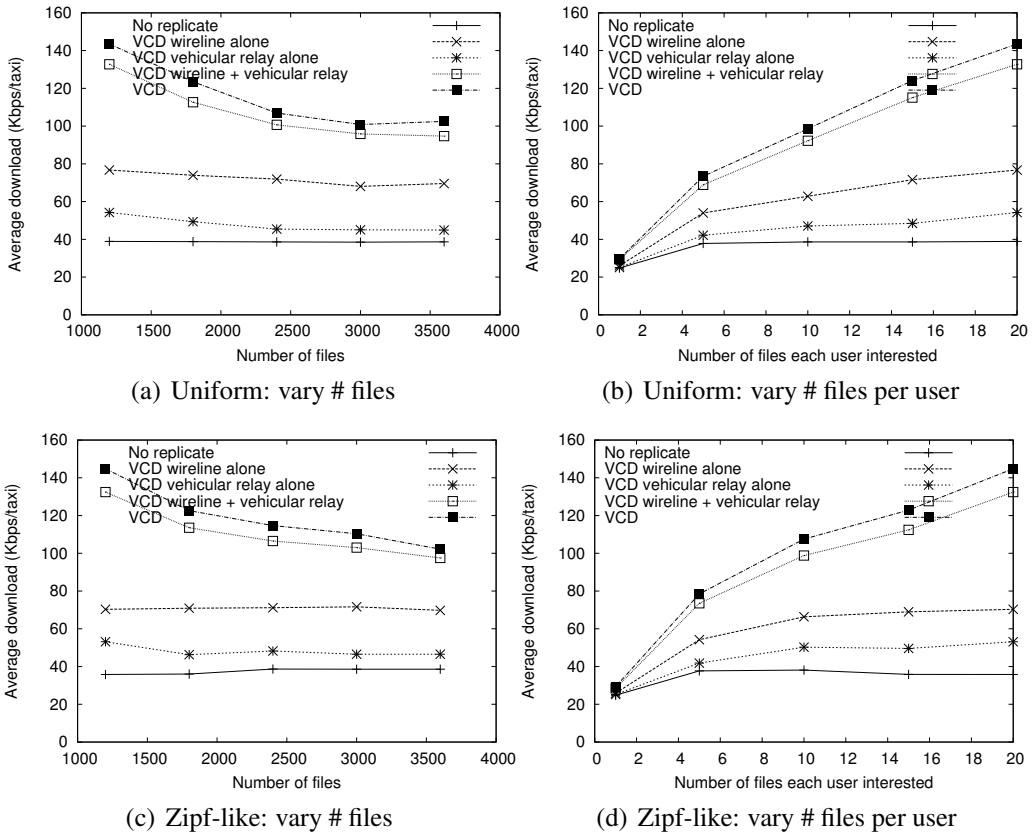
from the traces. Figure 5.7 summarizes the performance results. We make the following observations. First, VCD continues to perform the best in all cases. Second, increasing the number of vehicles initially improves the average throughput because more content are available locally at APs due to previous requests coming from other users. In addition, increasing the number of vehicles also creates more wireless relay opportunities. However, a further increase degrades performance due to increased contention for limited wireline and wireless resources. Third, the benefit of mesh replication increases with the number of vehicles. When we use all the vehicles in the two-hour traces, we find that the mesh replication helps to increase throughput by 17-22%. This is because increasing the number of vehicles increases vehicular relay opportunities and makes it more likely to have content available at nearby mesh nodes.

Varying traffic demands: Figure 5.8 shows the performance for uniformly and Zipf-like distributed traffic demand, respectively. As before, VCD performs the best in all cases. The performance of uniform and Zipf-like distributed traffic receives similar performance. Moreover, decreasing the total number of files tends to improve performance as demands are more concentrated and less replication is

required to satisfy them. Finally, the replication benefit tends to increase with an increasing number of files requested by each user. This is because when a user is interested in more content, it is more likely to have some locally available content that satisfies the user.

5.7 Trace-Driven Emulation of VCD

The goal of our Emulab implementation is twofold: (1) validate simulation results, and (2) evaluate the performance of VCD at scale, which is hard to do in testbed experiments.



(a) Uniform: vary # files

(b) Uniform: vary # files per user

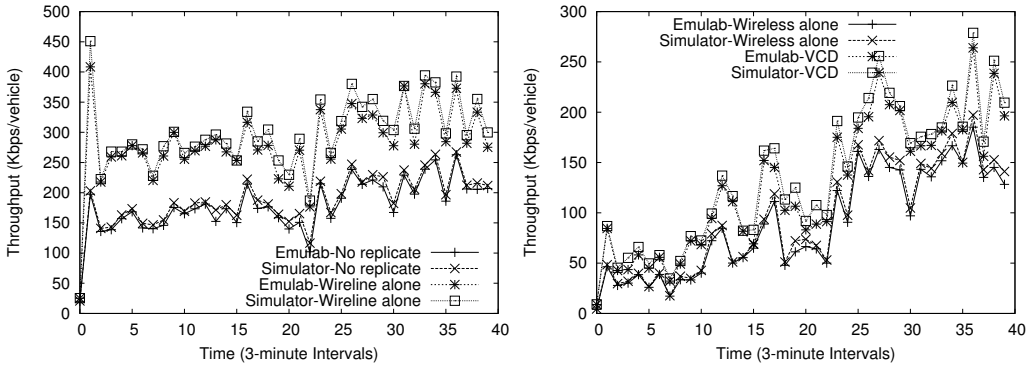
(c) Zipf-like: vary # files

(d) Zipf-like: vary # files per user

Figure 5.8: Average throughput under varying traffic demands (San Francisco, vehicle=50, range=100m, coffee shops).

Validation: To validate the simulation results, we compare them against those obtained from Emulab under identical settings. We consider the 30 most interactive APs from the trace contacting 100 vehicles. The radio range is 200m. Given limited machine availability on Emulab, we emulate multiple APs and vehicles on each machine. This limits the link capacity we can select per AP or per vehicle. Hence, our evaluation uses 1Mbps and 6Mbps as the Internet and wireless link capacities, respectively.

Figure 5.9 shows the average throughput for each interval in Emulab and simulator. In Figure 5.9(a), we consider that all APs have Internet connectivity and compare the simulation and emulation performance under no replication and wireline replication alone. We observe that the simulation results closely follow that of Emulab and the discrepancy between them is below 10%. Next we consider only 10% of the APs have Internet connectivity and compare the performance for vehicular replication alone and VCD in both simulator and Emulab. In this case, since most APs are not connected to the Internet and there is no mesh connectivity, most content is replicated via vehicles. Figure 5.9 (b) shows that the simulation results match well with Emulab results: within 10% difference for both vehicular



(a) All APs have Internet (b) 10% APs have Internet

Figure 5.9: Cross validation: comparing performance in Emulab and simulation

Packet type	Avg KB	% of total traffic
Controller to APs	192	0.006
APs to controller	1483	0.048
Content server to AP data	3078200	99.946
Vehicles to APs	49122	1.599
APs to vehicles data	3023100	98.401

Table 5.2: Average control message overhead per interval.

replication and VCD.

Micro-benchmarks: The following micro-benchmark results show that our implementation is efficient and light-weight even when operating at scale. We emulate the 120 most interactive APs and 317 vehicles from the trace.

Table 5.2 shows the per-interval control message overhead. We observe that control messages constitute only 0.054% of the total wireline traffic exchanged amongst APs and between APs and the controller, and constitute only 1.6% of the total wireless traffic between APs and vehicles.

Controller efficiency: We need to ensure that the centralized controller does not become the performance bottleneck. Next we evaluate the efficiency of the controller. On a 2.133GHz Xeon machine with 3GB RAM, average CPU and memory utilization of the controller is 2% and 38 MB respectively. The average latency at the controller is 7.8s, which is a small fraction of the 3-minute interval. Table 5.3 further shows the breakdown of the processing latency at the controller. The pre-processing stage involves predicting which APs will be visited and preparing input

	Average Latency (ms)
Pre-processing for LP	1307
LP Computation	6512
LP Result Processing	32
Total	7851

Table 5.3: Average controller processing delay per interval

file for lp_solve. Out of 7.8s, the LP computation takes 6.5s. It is performed on Emulab using *lp_solve* [86] due to licensing issues with *cplex* [32], and the time can be further reduced if cplex is used instead.

AP load: Finally we evaluate the scalability of APs by running 120 instances of the AP on 2.133GHz Xeon machines with 3GB RAM. We find that all APs have roughly the same usage, with each AP instance consuming only 0.01% CPU load and 33 MB of memory. Therefore it is light-weight.

5.8 Testbed Experiments

We evaluate our approach using two testbeds to understand its feasibility and effectiveness under realistic wireless conditions. The first testbed consists of 9 APs deployed in office buildings near the road. The APs are Linux desktops equipped with 802.11b radios, which are set to a fixed data rate of 11Mbps. The second testbed consists of 4 APs deployed outdoor equipped with 802.11n radios that use auto-rate. 802.11n radios use 2.4GHz frequency with a 20MHz band. In both testbeds, the APs have 1Mbps wireline access link connecting to the back-end content server. In the 802.11b testbed, 3 out of the 9 APs forms a mesh network as a 2-hop linear chain, whereas the 4 APs in the 802.11n testbed forms a mesh network with pairwise connectivity. In both testbeds, mesh communication takes place using additional 802.11b radios. We implement clients on both Windows Mobile Smartphones and Windows XP Laptops. Smartphone clients are used in 802.11b experiments and laptop clients are used in 802.11n experiments. Both clients ran a video streaming application during the car ride. The cars travelled around the testbed at 15 mph (speed limit). We expect that the driving speed does not significantly affect the performance when association time is small, because increasing speed reduces both on-time (*i.e.*, contact time) and off-time (*i.e.*, the

time between two consecutive contacts).

Connection setup: Due to deployment constraints, the placement of our 802.11b APs is not ideal: 4 of our APs were placed on the 3rd floor of buildings, limiting their range; and 3 APs were placed in high AP density areas, with 50-70 APs within their range, causing heavy interference. This deployment stress-tests our system. In our experiments during car rides, we were able to associate successfully for 65.2% of all attempts. Most of the failures came from the 3 APs deployed in the high AP density area: association success percentage was only 33.3% for these APs. In fact, even the Windows Mobile Wi-Fi manager utility experienced problems such as very long connection time and adapter freezing near these APs even without any movement. The other access points can successfully associate for 85.7% of the time. The association time in our experiments has minimum, median and maximum of 36ms, 844ms, and 14867ms, respectively. 70% of the associations finish within 2 seconds. We retry association up to 7 times and the median retry count is 1.

In our 802.11n outdoor testbed, association success rate was 89.58% out of 48 attempts. The minimum, median and maximum association times were 48 ms, 162 ms, and 4086 ms, respectively. 80% of the associations finish within 246 ms and the median retry count was 1. The better results for 802.11n testbed were because (i) we used laptops as clients, (ii) APs were placed outdoor closer to vehicles, and (iii) MIMO in 802.11n improves received signal strength.

Wireline and mesh replication: We implemented a video streaming application that can play H.264 videos (downloaded from APs) encoded at 64Kbps. We divide every video into multiple files and use network coding to generate random linear combination of packets within a file. Once enough packets are received for the file, the file is decoded and passed to the video player on the smartphone/laptop to play in proper order using the Windows Mobile media player plugin.

	Download (kB)	Play time (sec)
No replication	29297	3662
Wireline	71930	8991
Wireline + Mesh	79440	9930
Full replication	92493	11562

Table 5.4: Throughput of wireline and mesh replication in the 802.11b testbed

	Download (kB)	Play time (sec)
No replication	16857	2107
Wireline	123175	15387
Wireline + Mesh	130827	16353
Full replication	136479	17060

Table 5.5: Throughput of wireline and mesh replication in the 802.11n testbed

Tables 5.4 and 5.5 compare the performance of our optimized wireline and mesh replication with no replication and full replication at all the APs in 802.11b and 802.11n testbeds, respectively. We consider two performance metrics: total download size and total amount of time the video can play (which is proportional to the download size). We report the averages over 3 runs. The full replication assumes every AP has all the files and serves as an upper bound. In both experiments, we follow the planned trajectory, which was fed as input to the controller. In 802.11b testbed, wireline replication alone and wireline plus mesh replication performs 2.45x and 2.7x that of no replication, respectively. In 802.11n testbed, the throughput of wireline and wireline plus mesh replication is 7.3x and 7.8x that of no replication, respectively. This demonstrates the effectiveness of replication. Moreover, the benefit increases with wireless capacity. There is a gap between the performance of VCD and full replication, since the Internet bottleneck prevents complete replication of all the required files.

Vehicular replication: To show the benefit of vehicular replication, we use the following setup. Car 1 follows the route $AP1 - AP2$, and Car 2 follows the route $AP2 - AP1$. Car 1 possesses files 1-20 and is interested in files 21-40, while car 2

has files 21-40 and is interested in files 1-20. Both $AP1$ and $AP2$ lack Internet and mesh connectivity. Therefore, without vehicular replication, neither car can get the content it is interested in and the total throughput is 0 under no replication, wireline replication alone, and mesh replication alone.

In comparison, VCD exploits the vehicular replication opportunity. When car 1 meets $AP1$, VCD finds that files 1-20 have highest utility because it predicts car 2 will visit $AP1$ soon and need these files. So $AP1$ instructs the car to upload them first. Similarly, car 2 uploads file 21-40 at $AP2$. When car 1 reaches $AP2$ it can download these files. Similarly, car 2 can download files 1-20 from $AP1$, leading to much higher throughput. Table 5.6 shows that both cars download their interested files in the actual road experiments.

5.9 Summary

In this chapter, we present the VCD system that provides high-bandwidth content access to vehicular passengers by utilizing opportunistic connections to Wi-Fi access points along the road. VCD predicts which APs a vehicle will encounter in the future and proactively pushes content to these APs by leveraging both wireline and wireless connectivity. Using trace-driven simulation and Emulab-based emulation, we show that VCD is capable of downloading 3-6X as much content as no replication and 2-4X as much content as wireline or vehicular replication alone. The benefit further increases as the ratio between wireless and wireline capacity

	No replication		Wireless replication	
	Car 1	Car 2	Car 1	Car 2
AP1	0	0	Upload 780 pkts	Download 780 pkts,20 files
AP2	0	0	Download 1159 pkts, 20 files	Upload 1159 pkts

Table 5.6: Comparison between performance with and without vehicular replication.

increases. We further develop a full-fledged prototype of VCD using two testbeds: a 9-AP 802.11b testbed and a 4-AP 802.11n testbed. Our experience suggests that VCD is an effective approach for vehicular content distribution.

Chapter 6

Conclusion

In this dissertation, we emphasize the importance of opportunistic communication in wireless mesh networks and vehicular networks. We present a series of novel optimization frameworks that enable systematic optimization of opportunistic communication in wireless mesh networks and vehicular networks.

First, we present an accurate model-driven optimization framework for opportunistic routing in IEEE 802.11 mesh networks. This includes an accurate interference model, a general algorithm that jointly optimizes routes and rate-limits, an iterative optimization algorithm to cope with the non-convex model, and a practical routing protocol. Through testbed implementation and simulation, we validate the accuracy of our model and shows that the performance of our protocol excels the state-of-the-art routing protocols. Furthermore, we show that our approach is robust against inaccuracy introduced by dynamic network conditions.

Second, we present an overlay-based optimization framework that effectively incorporates inter-flow coding into opportunistic routing. The framework uses a novel hierarchical approach where overlay network performs inter-flow coding to reduce the traffic in the network and underlay network performs opportunistic routing to reliably transfer the traffic of the overlay. By Qualnet simulation, we show that our overlay-based optimization approach is effective and brings significant performance improvement in opportunistic routing.

Third, we present an interval-based optimization framework that optimizes

replication schemes to enable high-bandwidth content distribution in vehicular networks. We present an accurate mobility prediction algorithm and novel replication optimization schemes that effectively take advantage of Internet connectivity, local wireless connectivity, and vehicular relay connectivity. Using trace-driven simulation and Emulab-based emulation, we show our replication scheme brings significant performance improvement. We further develop a full-fledge prototype VCD system in testbed that supports real video streaming running on smart phones and laptop clients. Our testbed results also confirm the effectiveness of our design.

Future Work: We plan to enhance our framework in the following aspects:

First, we plan to further enhance the robustness of our optimization framework against traffic and topology variations. We can extend various traffic engineering techniques developed in the Internet to optimize wireless networks. In particular, a traffic engineering system usually collects a set of traffic matrices and uses their convex combination to cover the space of common traffic patterns for optimization. These new demand constraints are compact and can be easily incorporated into our framework. We plan to extend this technique to cope with both traffic and topology variations in wireless networks.

Second, we plan to enhance the scalability of our framework as solving the optimization in a centralized fashion becomes unscalable with the increasing network size. For example, we can apply decomposition techniques developed for distributed convex optimization (*e.g.*, [68]) to solve the optimization in a fully distributed fashion.

Third, we plan to incorporate diverse factors such as user preferences into our optimization objectives. WiFi technology and 3G/4G provides different bandwidth at a different cost. While WiFi provides intermittent high-bandwidth connec-

tivity, 3G/4G allows always-on but lower-bandwidth and/or high-cost path. We can leverage both types of network connectivity to enhance performance.

Bibliography

[1] A. Abutaleb and V. O. K. Li. Paging strategy optimization in personal communication systems. *Wireless Networks*, 3(3):195–204, 1997.

[2] S. Agarwal, J. Padhye, V. N. Padmanabhan, L. Qiu, A. Rao, and B. Zill. Estimation of link interference in static multi-hop wireless networks. In *Proc. of IMC*, 2005.

[3] Ahlswede, Cai, Li, and Yeung. Network information flow. *IEEETIT: IEEE Transactions on Information Theory*, 46, 2000.

[4] N. Ahmed, U. Ismail, S. Keshav, and K. Papagiannaki. Online estimation of RF interference. In *Proc. of ACM CoNext*, Dec. 2008.

[5] N. Ahmed and S. Keshav. Smarta: A self-managing architecture for thin access points. In *Proc. of ACM CoNEXT*, Dec. 2006.

[6] I. F. Akyildiz and W. Wang. The predictive user mobility profile framework for wireless multimedia networks. *IEEE/ACM Trans. Netw.*, 12(6):1021–1035, 2004.

[7] A. R. Aljadhai and T. Znati. Predictive mobility support for qoS provisioning in mobile wireless environments. *IEEE Journal on Selected Areas in Communications*, 19(10):1915–1930, 2001.

[8] AT&T DSL. `http://www.att.com/gen/general?pid=6431`.

[9] AWE Mesh Router. http://www.nomadio.net/AWE_overview.pdf.

[10] B. Awerbuch, D. Holmer, and H. Rubens. High throughput route selection in mult-rate ad hoc wireless networks. In *Technical report, Johns Hopkins University*, 2003.

[11] A. Balasubramanian, B. Levine, and A. Venkataramani. DTN routing as a resource allocation problem. In *Proc. of SIGCOMM*, pages 373–384, 2007.

[12] A. Balasubramanian, B. Levine, and A. Venkataramani. Enhancing interactive web applications in hybrid networks. In *Proc. of MobiCom*, Sept. 2008.

[13] A. Balasubramanian, R. Mahajan, A. Venkataramani, B. Levine, and J. Zahorjan. Interactive WiFi connectivity for moving vehicles. In *Proc. of SIGCOMM*, 2008.

[14] A. Balasubramanian, Y. Zhou, W. B. Croft, B. N. Levine, and A. Venkataramani. Web search from a bus. In *CHANTS '07: Proceedings of the second ACM workshop on Challenged networks*, pages 59–66, New York, NY, USA, 2007. ACM.

[15] N. Banerjee, M. Corner, D. Towsley, and B. Levine. Relays, base stations, and meshes: Enhancing mobile networks with infrastructure. In *Proc. of MobiCom*, Sept. 2008.

[16] A. Bhattacharya and S. K. Das. Lezi-update: an information-theoretic framework for personal mobility tracking in pcs networks. *Wirel. Netw.*, 8(2/3):121–135, 2002.

[17] G. Bianchi. Performance analysis of the IEEE 802.11 distributed corrdination function. *IEEE Journal on Selected Areas in Communications*, Mar. 2000.

[18] S. Biswas and R. Morris. ExOR: opportunistic multi-hop routing for wireless networks. In *Proc. of ACM SIGCOMM*, Aug. 2005.

[19] BMW car2car communication development. `http://www.motorauthority.com/bmw-enlists-in-car-2-car-communications-%development.html`.

[20] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine. MaxProp: routing for vehicle-based disruption-tolerant networks. In *Proc. of IEEE INFOCOM*, pages 1–11, Apr. 2006.

[21] B. Burns, O. Brock, and B. N. Levine. Mv routing and capacity building in disruption tolerant networks. In *In Proc. IEEE INFOCOM*, pages 398–408, 2005.

[22] V. Bychkovsky, B. Hull, A. Miu, H. Balakrishnan, and S. Madden. A measurement study of vehicular Internet access using in situ Wi-Fi networks. In *Proc. of MobiCom*, pages 50–61, 2006.

[23] Cabspotting. `http://www.cabspotting.com`.

[24] J. Camp and E. Knightly. Modulation rate adaptation in urban and vehicular environments: Cross-layer implementation and experimental evaluation. In *Proc. of MobiCom*, Sept. 2008.

[25] Car2Car communication consortium. `http://www.car-to-car.org`.

[26] Cartel. http://cartel.csail.mit.edu/doku.php.

[27] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading structure for randomness in wireless opportunistic routing. In *Proc. of SIGCOMM*, 2007.

[28] P. Chaporkar and A. Proutiere. Adaptive network coding and scheduling for maximizing throughput in wireless networks. In *Proc. of ACM MobiCom*, 2007.

[29] B. B. Chen and M. C. Chan. MobTorrent: a framework for mobile internet access from vehicles. In *Proc. of IEEE INFOCOM*, 2009.

[30] Click. http://pdos.csail.mit.edu/click/.

[31] D. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proc. of ACM MobiCom*, 2003.

[32] Cplex. http://www.ilog.com/products/cplex/.

[33] S. Das, A. Nandan, and G. Pau. Spawn: a swarming protocol for vehicular ad-hoc wireless networks. In *Proc. of VANET*, 2004.

[34] S. Das, Y. Wu, R. Chandra, and C. Hu. Context based routing: Technique, applications and experience. In *Proc. of NSDI*, 2008.

[35] J. A. Davis, A. H. Fagg, and B. N. Levine. Wearable computers as packet transport mechanisms in highly-partitioned ad-hoc networks. In *ISWC*, pages 141–148. IEEE Computer Society, 2001.

[36] P. Deshpande, A. Kashyap, C. Sung, and S. Das. Predictive methods for improved vehicular wifi access. In *Proc. of ACM MobiSys*, 2009.

[37] P. Deshpande, A. Kashyap, C. Sung, and S. R. Das. Predictive methods for improved vehicular wifi access. In K. Zielinski, A. Wolisz, J. Flinn, and A. LaMarca, editors, *MobiSys*, pages 263–276. ACM, 2009.

[38] R. Draves, J. Padhye, and B. Zill. Routing in multi-radio, multi-hop wireless mesh networks. In *Proc. of ACM MobiCom*, Sept. - Oct. 2004.

[39] R. Dube, C. Rais, K.-E. Wang, and S. Tripathi. Signal stability based adaptive routing (SSA) for ad-hoc mobile networks. In *IEEE Personal Comm*, Feb. 1997.

[40] R. Dube, C. D. Rais, K.-Y. Wang, and S. K. Tripathi. Signal stability based adaptive routing (SSA) for ad-hoc mobile networks. Technical Report CS-TR-3646, University of Maryland, College Park, Aug. 1996.

[41] H. Dubois-Ferrier, M. Grossglauser, and M. Vetterli. Least-cost opportunistic routing. In *Proc. of Allerton*, Sept. 2007.

[42] Emulab. `http://www.emulab.net`.

[43] J. Eriksson, H. Balakrishnan, and S. Madden. Cabernet: Vehicular content delivery using WiFi. In *Proc. of MobiCom*, Sept. 2008.

[44] H. Feng, Y. Shu, S. Wang, and M. Ma. SVM-based models for predicting WLAN traffic. In *Proc. of IEEE ICC*, 2006.

[45] R. H. Frenkiel, B. R. Badrinath, J. Borrs, J. B. As, and R. D. Yates. The infostations challenge: Balancing cost and ubiquity in delivering wireless data. *IEEE Personal Communications*, 7:66–71, 1999.

[46] Y. Gao, J. Lui, and D. M. Chiu. Determining the end-to-end throughput capacity in multi-hop networks: Methodolgy and applications. In *Proc. of ACM SIGMETRICS*, Jun. 2006.

[47] M. Garetto, T. Salonidis, , and E. Knightly. Modeling per-flow throughput and capturing starvation in CSMA multi-hop wireless networks. In *Proc. of IEEE INFOCOM*, March 2006.

[48] M. Garetto, J. Shi, and E. Knightly. Modeling media access in embedded two-flow topologies of multi-hop wireless networks. In *Proc. of ACM MobiCom*, Aug. - Sept. 2005.

[49] M. Gastpar and M. Vetterli. On the capacity of wireless networks: the relay case. In *Proc. of IEEE INFOCOM*, Jun. 2002.

[50] J. Ghosh, M. J. Beal, H. Q. Ngo, and C. Qiao. On profiling mobility and predicting locations of wireless users. In *Proc. of REALMAN*, pages 55–62, 2006.

[51] Firetide wireless mesh brings rural korean communities into the network fold. `http://www.reuters.com/article/pressRelease/idUS111778+02-Sep-2008+BW20%080902`.

[52] M. Grossglauser and D. N. C. Tse. Mobility increases the capacity of ad hoc wireless networks. In *Proc. of IEEE INFOCOM*, April 2001.

[53] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2), Mar. 2000.

[54] D. Hadaller, S. Keshav, T. Brecht, and S. Agarwal. Vehicular opportunistic communication under the microscope. In *Proc. of MobiSys*, pages 206–219, 2007.

[55] R. Haemmerle. LP solve – incremental version. `http://contraintes.inria.fr/haemmerl/lp_solve_inc/`.

[56] T. Ho, M. Medard, J. Shi, M. Effros, and D. R. Karger. On randomized network coding. In *In Proceedings of 41st Annual Allerton Conference on Communication, Control, and Computing*, 2003.

[57] T. Ho, M. Medard, J. Shi, M. Eros, and D. R. Karger. On randomized network coding, Oct. 06 2003. `web.mit.edu/trace/www/allerton.pdf`.

[58] Honda to exhibit its latest advanced safety vehicles. `http://world.honda.com/news/2009/4090219ITS-Safety-2010/`.

[59] Jaggi, Sanders, Chou, Effros, Egner, Jain, and Tolhuizen. Polynomial time algorithms for multicast network code construction. *IEEETIT: IEEE Transactions on Information Theory*, 51, 2005.

[60] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Mdard. Resilient network coding in the presence of byzantine adversaries. In *in Proc. 26th Annual IEEE Conf. on Computer Commun., INFOCOM*, pages 616–624, 2007.

[61] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu. Impact of interference on multi-hop wireless network performance. In *Proc. ACM MobiCom*, 2003.

[62] S. Jain, M. J. Demmer, R. K. Patra, and K. R. Fall. Using redundancy to cope with failures in a delay tolerant network. In R. Guérin, R. Govindan, and G. Minshall, editors, *SIGCOMM*, pages 109–120. ACM, 2005.

[63] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 145–158, New York, NY, USA, 2004. ACM.

[64] D. B. Johnson, D. A. Maltz, and J. Broch. DSR: The dynamic source routing protocol for multihop wireless ad hoc networks. In *Ad Hoc Networking*, 2001.

[65] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. In *Proc. of ACM ASPLOS*, Oct. 2002.

[66] A. Kashyap, S. Das, and S. Ganguly. A measurement-based approach to modeling link capacity in 802.11-based wireless networks. In *Proc. of ACM MobiCom*, Sept. 2007.

[67] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft. XORs in the air: Practical wireless network coding. In *Proc. of SIGCOMM*, Sept. 2006.

[68] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan. Rate control in communication networks: Shadow prices, proportional fairness and stability. *Jounal of the Operational Research Society*, 1998.

[69] T. Kim, S. Vural, and I. Broustis. A framework for joint network coding and transmission rate control in wireless networks. In *Proc. of INFOCOM*, 2010.

[70] R. Koetter and M. Médard. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, 11(5):782–795, Oct. 2003.

[71] D. Koutsonikolas, Y. C. Hu, and C. Wang. Xcor: Synergistic interflow network coding and opportunistic routing. In *MOBICOM Student Research Comptition (SRC)*, 2008.

[72] D. Koutsonikolas, C.-C. Wang, and Y. C. Hu. CCACK: efficient network coding based opportunistic routing through cumulative coded acknowledgments. In *Proc. of IEEE INFOCOM*, 2010.

[73] A. Kumar, E. Altman, D. Miorandi, and M. Goyal. New insights from a fixed point analysis of single cell IEEE 802 .11 wireless LANs. In *Proc. of IEEE INFOCOM*, Mar. 2005.

[74] J. Le, J. Lui, and D. Chiu. DCAR: distributed coding-aware routing in wireless networks. *Transactions of Mobile Computing*, 2010.

[75] K. C. Lee, S.-H. Lee, R. Cheung, U. Lee, and M. Gerla. First Experience with CarTorrent in a Real Vehicular Ad Hoc Network Testbed. In *MOVE*, 2007.

[76] J. Leguay, T. Friedman, and V. Conan. Dtn routing in a mobility pattern space. In *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 276–283, New York, NY, USA, 2005. ACM.

[77] Li, Yeung, and Cai. Linear network coding. *IEEETIT: IEEE Transactions on Information Theory*, 49, 2003.

[78] J. Li, C. Blake, D. S. J. D. Couto, H. I. Lee, and R. Morris. Capacity of ad hoc wireless networks. In *Proc. of MobiCom*, Jul. 2001.

[79] Y. Li, L. Qiu, Y. Zhang, R. Mahajan, and E. Rozner. Predictable performance optimization for wireless networks. In *Proc. of ACM SIGCOMM*, Aug. 2008.

[80] Y. Li, L. Qiu, Y. Zhang, R. Mahajan, Z. Zhong, G. Deshpande, and E. Rozner. Effects of interference on throughput of wireless mesh networks: Pathologies and a preliminary solution. In *Proc. of HotNets-VI*, Nov. 2007.

[81] B. Liang and Z. J. Haas. Predictive distance-based mobility management for PCS networks. In *Proc. of INFOCOM*, Apr. 1999.

[82] Y. Lin, B. Li, and B. Liang. CodeOR: opportunistic routing in wireless mesh networks with segmented network coding. In *Proc. of IEEE ICNP*, Oct. 2008.

[83] Y. Lin, B. Liang, and B. Li. SlideOR: online opportunistic network coding in wireless mesh networks. In *Proc. of IEEE INFOCOM*, Mar. 2010.

[84] A. Lindgren, A. Doria, and O. Schelén. Probabilistic routing in intermittently connected networks. *Mobile Computing and Communications Review*, 7(3):19–20, 2003.

[85] T. Liu, P. Bahl, S. Member, and I. Chlamtac. Mobility modeling, location tracking, and trajectory prediction in wireless ATM networks, Mar. 03 1998.

[86] Lp_solve: Linear programming code. http://www.cs.sunysb.edu/~algorith/implement/lpsolve/implement.shtml.

[87] M.-H. Lu, P. Steenkiste, and T. Chen. Design, implementation, and evaluation of an efficient opportunistic retransmission protocol. In *Proc. of ACM MobiCom*, 2009.

[88] D. Lun, M. Medard, and R. Koetter. Network coding for efficient wireless unicast. *International Zurich Seminar on Communications*, 2006.

[89] D. S. Lun, M. Medard, and R. Koetter. Efficient operation of wireless packet networks using network coding. In *International Workshop on Convergent Technologies (IWCT)*, 2005.

[90] Madwifi. http://madwifi.org.

[91] Mazda advanced safety vehicle 4 to begin public road trials. http://www.gizmag.com/mazda-asv4-road-trials/8862/.

[92] L. McNamara, C. Mascolo, and L. Capra. Media sharing based on colocation prediction in urban transport. In *Proc. of MobiCom*, pages 58–69, 2008.

[93] Meraki MR58. http://meraki.com/products_services/access_points/MR58/.

[94] Mitsubishi fuso develops advanced safety vehicle fuso asv-4 to join its-safety2010 in japan. http://www.mitsubishi-fuso.com/en/press/090129/090129.html.

[95] A. K. Miu, H. Balakrishnan, and C. E. Koksal. Improving loss resilience with multi-radio diversity in wireless networks. In *In Proc. of ACM MOBICOM*, 2005.

[96] A. K. Miu, H. Balakrishnan, and C. E. Koksal. Improving loss resilience with multi-radio diversity in wireless networks. In *Proc. of ACM MobiCom*, 2005.

[97] A. K. Miu, G. Tan, H. Balakrishnan, and J. Apostolopoulos. Divert: Fine-grained path selection for wireless lans. In *Proc. of ACM MobiSys*, 2004.

[98] Mobile Broadband Review 2010. `http://mobile-broadband-services-review.toptenreviews.com/`.

[99] MORE source code. `http://people.csail.mit.edu/szym/more/README.html`.

[100] J. Moy. OSPF version 2. In *Internet Engineering Task Force, RFC 2328*, Apr. 1998. http://tools.ietf.org/html/rfc2328.

[101] S. Murthy and J. J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *Mob. Netw. Appl.*, 1(2):183–197, 1996.

[102] V. Navda, A. P. Subramanian, K. Dhanasekaran, A. Timm-Giel, and S. Das. Mobisteer: using steerable beam directional antenna for vehicular network access. In *Proc. of MobiSys*, pages 192–205, 2007.

[103] A. J. Nicholson and B. D. Noble. Breadcrumbs: Forecasting mobile connectivity. In *Proc. of MobiCom*, Sept. 2008.

[104] D. Niculescu. Interference map for 802.11 networks. In *Proc. of IMC*, 2007.

[105] L. M. S. C. of the IEEE Computer Society. Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE Standard 802.11*, 1999.

[106] J. Ott and D. Kutscher. Drive-thru internet: Ieee 802.11b for "automobile" users. volume 1, page 373, 2004.

[107] J. Ott and D. Kutscher. A disconnection-tolerant transport for drive-thru internet environments. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1849–1862 vol. 3, 2005.

[108] J. Ott and D. Kutscher. Exploiting regular hot-spots for drive-thru internet. In *in Proceedings of KiVS 2005*, 2005.

[109] V. N. Padmanabhan and J. C. Mogul. Using predictive prefetching to improve world wide web latency. *SIGCOMM Comput. Commun. Rev.*, 26(3), 1996.

[110] Park. Codecast: A network-coding based ad hoc multicast protocol.

[111] V. D. Park and M. S. Corson. Temporally-ordered routing algorithm. In *Internet draft, draft-ietf-manet-tora-spec-01.txt*, August 1998.

[112] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *Proc. of ACM SIGCOMM*, Aug.-Sept. 1994.

[113] C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *Proc. of the Workshop on Mobile Computing Systems and Applications*, Feb. 1999.

[114] Laptop & smartphone users prefer Wi-Fi to 3g; willing to pay for citywide Wi-Fi. http://www.teleclick.ca/2009/02/laptop-willing-to-pay-for-c
i/.

[115] C. Qin, Y. Xian, C. Gray, N. Santhapuri, and S. Nelakuditi. I2mix: Integration of intra-ow and inter-ow wireless network coding. In *In Proc. of IEEE Workshop on Wireless Network Coding (WiNC)*, 2008.

[116] L. Qiu, Y. Zhang, F. Wang, M. K. Han, and R. Mahajan. A general model of wireless interference. In *Proc. of ACM MobiCom*, Sept. 2007.

[117] The Qualnet simulator from Scalable Networks Inc. http://www.scalable-networks.
com/.

[118] B. Radunovic, C. Gkantsidis, P. Key, and P. Rodriguez. An optimization framework for opportunistic multipath routing in wireless mesh networks. In *Proc. of IEEE INFOCOM*, April 2008.

[119] B. Randunovi and J. Y. L. Boudec. Rate performance objectives of multihop wireless networks. In *Proc. of IEEE INFOCOM*, Apr. 2004.

[120] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Measurement-based models of delivery and interference. In *Proc. of ACM SIGCOMM*, 2006.

[121] MIT Roofnet. http://www.pdos.lcs.mit.edu/roofnet/.

[122] E. Rozner, M. K. Han, L. Qiu, and Y. Zhang. Model-driven optimization of opportunistic routing. Technical Report TR-11-12, The University of Texas at Austin, Dept. of Computer Science, Austin, TX, 2010.

[123] T. Salonidis, M. Garetto, A. Saha, and E. Knightly. Identifying high throughput paths in 802.11 mesh networks: a model-based approach. In *Proc. of IEEE ICNP*, Oct. 2007.

[124] B. Scheuermann, W. Hu, and J. Crowcroft. Near-optimal coordinated coding in wireless multihop networks. In *Proc. of CoNEXT*, 2007.

[125] Seattle Bus Traces. `http://crawdad.cs.dartmouth.edu/meta.php?name=rice/ad_hoc_city`.

[126] Seattle wireless. `http://www.seattlewireless.net`.

[127] S. Sengupta, S. Rayanchu, and S. Banerjee. An analysis of wireless network coding for unicast sessions: The case for coding-aware routing. In *Proc. of IEEE INFOCOM*, Apr. 2007.

[128] S. Sengupta, S. K. Rayanchu, and S. Banerjee. An analysis of wireless network coding for unicast sessions: The case for coding-aware routing. In *INFOCOM*. IEEE, 2007.

[129] J. K. Shapiro, D. Towsley, and J. Kurose. Optimization-based congestion control for multicast communications. *IEEE Communication Magazine*, 2002.

[130] T. Small and Z. J. Haas. Resource and performance tradeoffs in delay-tolerant wireless networks. In *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 260–267, New York, NY, USA, 2005. ACM.

[131] I. C. Society. IEEE standard for information technology - telecommunications and information exchange between systems - local and metropolitan

area networks - specific requirements 802.11r part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 2: Fast basic service set (bss) transition. *IEEE Standard 802.11*, 2008.

[132] F. Soldo, A. Markopoulou, and A. Toledo. A simple optimization model for wireless opportunistic routing with intra-session network coding. In *Proc. of IEEE NetCod*, Jun. 2010.

[133] L. Song, D. Kotz, R. Jain, and X. He. Evaluating location predictors with extensive Wi-Fi mobility data. In *Proc. of INFOCOM*, Mar. 2004.

[134] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and wait: An efficient routing scheme for intermittently connected mobile networks, Feb. 03 2009.

[135] K. Srinivasan, M. Jain, J. I. Choi, T. Azim, E. S. Kim, P. Levis, and B. Krishnamachari. The kappa factor: Inferring protocol performance using inter-link reception correlation. In *Proc. of ACM MobiCom*, Sept. 2010.

[136] Memory cards. `http://en.wikipedia.org/wiki/Comparison_of_memory_cards`.

[137] J. J. T. Ho and H. Viswanathan. On network coding and routing in dynamic wireless multicast networks. In *Proc. of Workshop on Information Theory and its Applications*, 2006.

[138] TFA-RICE Wireless Mesh tfa rice wireless mesh. `http:tfa.rice.edu`.

[139] Toyota and Honda vehicle-to-vehicle communication systems. `http://www.motorauthority.com/toyota-and-honda-start-testing-vehicle-to-vehicle-communications-systems.html`.

[140] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. *Wireless Networks*, 8(2-3):153–167, 2002.

[141] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical Report CS-200006, Duke University, 2000.

[142] J. Widmer and J.-Y. Le Boudec. Network coding for efficient communication in extreme networks. In *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 284–291, New York, NY, USA, 2005. ACM.

[143] Firetide mesh network connects workers on 2,275-ft dubai tower. `http://www.informationweek.com/story/showArticle.jhtml?articleID=204204%105&cid=RSSfeed_IWK_News`.

[144] Keyna data networks deploys strix system's mesh network in major cities. `http://www.allbusiness.com/media-telecommunications/telecommunications/%7754637-1.html`.

[145] Wifi-taxi. `http://www.wifi-taxi.com/cgv_en.htm`.

[146] City-wide Wi-Fi rolls out in UK. `http://news.bbc.co.uk/2/hi/technology/4578114.stm`.

[147] Cities unleash free Wi-Fi. `http://www.wired.com/gadgets/wireless/news/2005/10/68999`.

[148] Wikipedia. F1 score. `http://en.wikipedia.org/wiki/F1_score`.

[149] Worldwide pricelist for iPhone 3G Plans. `http://www.unwiredview.com/2008/07/16/worldwide-pricelist-for-iphone-3g%-plans/`.

[150] Yahoo! Local Search Web Services. `http://developer.yahoo.com/search/local/V3/localSearch.html`.

[151] Y. Yan, B. Zhang, H. T. Mouftah, and J. Ma. Practical coding-aware mechanism for opportunistic routing in wireless mesh networks. In *Proc. of ICC*, 2008.

[152] F. Yu and V. Leung. Mobility-based predictive call admission control and bandwidth reservation in wireless cellular networks. In *IEEE INFOCOM*, pages 518–526, 2001.

[153] Y. Yuan, H. Yuan, S. H. Wong, S. Lu, and W. Arbaugh. ROMER: resilient opportunistic mesh routing for wireless mesh networks. In *Proc. of IEEE WiMESH*, Sept. 2005.

[154] S. Yun and H. Kim. Rate diverse network coding: Breaking the broadcast bottleneck. In *Proc. of ACM MobiHoc*, 2010.

[155] K. Zeng, W. Lou, and H. Zhai. On end-to-end throughput of opportunistic routing in multirate and multihop wireless networks. In *Proc. of IEEE INFOCOM*, Apr. 2008.

[156] K. Zeng and Z. Y. W. Lou. Oportunistic routing in multi-radio multi-channel multi-hop wireless networks. In *Proc. of IEEE INFOCOM*, Mar. 2010.

[157] J. Zhang, Y. P. Chen, and I. Marsic. Network coding via opportunistic forwarding in wireless mesh networks. In *In Proc. of IEEE WCNC*, 2008.

[158] X. Zhang, J. Kurose, B. N. Levine, D. Towsley, and H. Zhang. Study of a Bus-Based Disruption Tolerant Network: Mobility Modeling and Impact

on Routing. In *Proc. ACM Annual Intl. Conf. on Mobile Computing and Networking (Mobicom)*, Montreal, Canada, September 2007.

[159] X. Zhang and B. Li. Optimized multipath network coding in lossy wireless networks. In *Proc. of ICDCS*, 2008.

[160] Z. Zhong and S. Nelakuditi. On the efficacy of opportunistic routing. In *Proc. of IEEE SECON*, Jun. 2007.